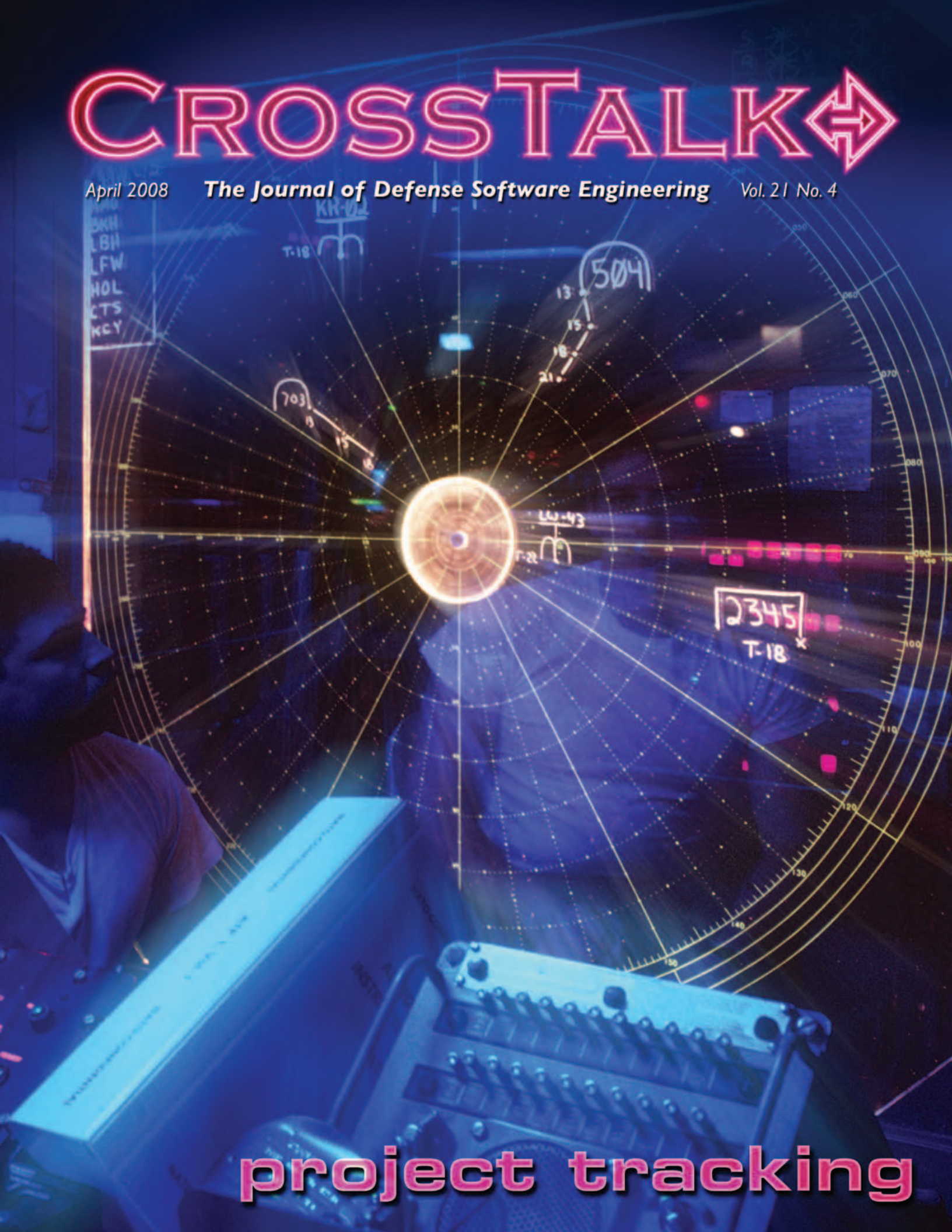

CROSSTALK
April 2008
The Journal of Defense Software Engineering
Vol. 21 No. 4
504
703
2345
T-18
project tracking

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
CrossTalk: The Journal of Defense Software Engineering. Volume 21, Number 4, April 2008	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **OO-ALC/MASE,6022 Fir Ave,Hill AFB,UT,84056-5820**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**32**	

Project Tracking

Software Engineering Technology

Open Forum

On The Cover

Cover Design by Kent Bingham

Additional art services provided by Janna Jensen

Departments

CrossTalk

Co-Sponsors:
DoD-CIO *The Honorable John Grimes*
OSD (AT&L) *Kristen Baldwin*
NAVAIR *Jeff Schwalb*
76 SMXG *Phil Perkins*
309 SMXG *Karl Rogers*
DHS *Joe Jarzombek*

Staff:
Managing Director *Brent Baxter*
Publisher *Elizabeth Starrett*
Managing Editor *Ken Davies*
Associate Editor *Chelene Fortier-Lozancich*
Article Coordinator *Nicole Kentta*
Phone (801) 775-5555
E-mail crosstalk.staff@hill.af.mil
CrossTalk Online www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the Department of Defense Chief Information Office (DoD-CIO); the Office of the Secretary of Defense (OSD) Acquisition, Technology and Logistics (AT&L); U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). DoD-CIO co-sponsor: Assistant Secretary of Defense (Networks and Information Integration). OSD (AT&L) co-sponsor: Software Engineering and System Assurance. USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG); and Ogden-ALC 309 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 18.

517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Keeping It Real

I got my first introduction to *real* project planning 10 years ago when Capability Maturity Model® Integration founding father Watts Humphrey came to Hill Air Force Base to pilot the Team Software Process℠ (TSP℠) with our TaskView project. As a project manager with more than 11 years of software experience, including three years in our software engineering process group and as a certified Personal Software Process℠ instructor, I was confident the plan I had led the TaskView team to construct was flawless. Before Watts' visit, we had spent several days defining all of the product components and used a modified Delphi approach to estimate the duration of each one. We had determined resources, made assignments, built a detailed Gantt chart with four dozen tasks, identified all the dependencies, planned for every milestone, and determined the critical path. We knew each and every deliverable, its customer, format, and need date. We were ready, or so I thought.

Over the next week, I watched as Watts worked painstakingly with our team members to create a *real* project plan, one that each engineer not only helped to create, but could use to guide his or her daily activities. Our meager four-dozen task Gantt chart was replaced by a more than 400-task Earned Value Plan, estimated both by a top-down and bottom-up approach; our risks were identified, recorded, categorized, prioritized, and assigned for follow-up, and a never-before-conceived-of quality plan was generated.

All of this was for a six-month project of six people ... and the results of this launch were staggering.

This plan was the basis of each weekly review. We were able to tell immediately when tasks were falling behind schedule. TaskView avoided or mitigated all of its critical risks. The quality of our product surpassed anything we had ever produced.

While I had known for several years that project planning and tracking were critical, it was not until this experience that I realized how useful these plans could be. Not only did they guide our actions, but they provided a basis for stability when requirements inevitably changed. In one case, for example, I was able to use the planning data to determine quantitatively that I could loan one of our engineers to another team without risking the TaskView delivery. Since that time, I have been a staunch advocate not only of the TSP but of taking the time to do *real* project planning. These solid, effective plans are worth all of the effort to create and maintain them.

This month's CrossTalk is filled with wonderful guidance for the critical tasks of planning and tracking software projects. First, in his article *Software Tracking: The Last Defense Against Failure*, software veteran Capers Jones details four *worst practices* leading to catastrophic failure and even litigation on software projects. With his usual prowess, Capers succinctly uncovers the mines in the minefield so that the rest of us can avoid them!

My two favorite articles in this month's issue are *Does Project Performance Stability Exist? A Re-examination of CPI and Evaluation of SPI(t) Stability* by Kym Henderson and Dr. Ofer Zwikael (spoiler alert: It does eventually), and Walt Lipke's *Schedule Adherence: A Useful Measure for Project Management*. Both articles focus on Lipke's new Earned Schedule measure: an exciting, innovative, and effective new way to make better use of Earned Value planning and tracking data.

As we all know, the most unpredictable portion of any software development or maintenance cycle is software testing. Dr. David J. Coe explains how to make testing more efficient and effective in *A Review of Boundary Value Analysis Techniques*.

Real project planning and tracking begins with making good estimates, and in the capstone article, *Truth and Confidence: Some of the Realities of Software Project Estimation*, Phillip G. Armour details the many issues that make software project estimating unusually difficult and suggests a fascinating new view of the process (and outcome) that makes estimates more usable.

To keep the attention of the *techies*, there is also an article from David Premeaux, discussing *VoIP Softphones*.

This month's articles will help guide us in making and following highly effective and *real* project plans.

David R. Webb
309th Software Maintenance Group

Software Tracking: The Last Defense Against Failure

Capers Jones
Software Productivity Research, LLC

From working as an expert witness in a number of lawsuits where large software projects were cancelled or did not operate correctly when deployed, I found that four major problems occur repeatedly: 1) accurate estimates are not produced or are overruled; 2) requirements changes are not handled effectively; 3) quality control is deficient; and 4) progress tracking fails to alert higher management to the seriousness of the issues. There are often other problems as well, but these four always occur in breach of contract litigation.

This article is based on software projects that were in litigation for breach of contract. It concentrates on four worst practices or the factors that most often lead to failure and litigation. A previous article dealt with additional problems noted during litigation [1].

For the purposes of this article, software failures are defined as software projects which met any of the following attributes:

1. Termination of project due to cost or schedule overruns.
2. Schedule or cost overruns in excess of 50 percent of initial estimates.
3. Applications which, upon deployment, fail to operate safely.
4. Lawsuits brought by clients for contractual non-compliance.

Although there are many factors associated with schedule delays and project cancellations, the failures that end up in court always seem to have four major deficiencies:

1. Accurate estimates were either not prepared or were rejected.
2. Change control was not handled effectively.
3. Quality control was inadequate.
4. Progress tracking did not reveal the true status of the project.

Let us consider each of these topics in turn.

Estimating Problems

Although cost estimation is difficult, there are a number of commercial software cost estimating tools that do a capable job: COCOMO II, KnowledgePlan, Price-S, SEER, SLIM, and SoftCost are examples.

However, just because an accurate estimate can be produced using a commercial estimating tool, this does not mean that clients or executives will accept it. In fact, from information presented during litigation, about half of the cases did not produce accurate estimates at all. The other half had accurate estimates but they were rejected and replaced by forced estimates based on business needs rather than team abilities.

The main reason that accurate estimates were rejected and replaced was the absence of supporting historical data. Without this, even accurate estimates may not be convincing. A lack of solid historical data makes project managers, executives, and clients blind to the realities of software development.

A situation such as this was one of the contributing factors to the long delay in opening the Denver International Airport. Estimates for the length of time to complete and debug the very complex baggage handling software were not believed [2].

For more than 60 years the software industry lacked a solid empirical foundation of measured results that was available to the public. Thus, almost every major software project is subject to arbitrary and sometimes irrational schedule and cost constraints. However, the International Software Benchmarking Standards Group (ISBSG), a non-profit organization, has started to improve this situation by offering schedule, effort, and cost benchmark reports to the general public[1]. Currently, more than 4,000 projects are available, and new projects are added at a rate of perhaps 500 per year.

There are other collections of software benchmark data, such as those gathered by the Gartner Group, David's Consulting Group, Software Productivity Research, and other companies, as well. However, this data is usually made available only on a subscription basis to specific clients of the organizations. The ISBSG data, by contrast, is available to the general public.

Changing Requirements

The average rate at which software requirements change is about 1 percent per calendar month. Thus, for a project with a 12 month schedule, more than 10 percent of the final delivery will not have been defined during the requirements phase. For a 36-month project, almost a third of the features and functions may have come in as an afterthought.

These are only average results. The author has observed a three-year project where the delivered product exceeded the functions in the initial requirements by about 289 percent. It is of some importance to the software industry that the rate at which requirements creep or grow can now be measured directly by means of the function point metric. This explains why function point metrics are now starting to become the basis of software contracts and outsource agreements.

Unfortunately, in projects where litigation occurred, requirements changes were numerous but their effects were not properly integrated into cost, schedule, and quality estimates. As a result, unplanned slippages and overruns occurred.

In several cases, the requirements changes had not been formally included in the contracts for development, and the clients refused to pay for changes that substantially affected the scope of the projects. One case involved 82 changes that totaled to more than 2,000 function points or about 20 percent of the original size of the initial requirements.

Since the defect potentials for changing requirements are larger than for the original requirements by about 10 percent, and since defect removal efficiency for changing requirements is lower by about 5 percent, projects with large volumes of changing requirements also have severe quality problems, which are usually invisible until testing begins. When testing begins, the project is in serious trouble because it is too late to bring the schedule and cost overruns under control.

Requirements changes will always occur for large systems. It is not possible to freeze the requirements of any real-world application and it is naïve to think it is possible. Therefore, leading companies are ready and able to deal with changes and do not let them become impediments to progress. For

projects developed under contract, the contract itself must include unambiguous language for dealing with changes.

Quality Problems

Effective software quality control is the most important single factor that separates successful projects from delays and disasters. The reason for this is because finding and fixing bugs is the most expensive cost element for large systems, and it takes more time than any other activity.

Successful quality control involves defect prevention, defect removal, and defect measurement activities. The phrase *defect prevention* includes all activities that minimize the probability of creating an error or defect in the first place. Examples of defect prevention activities include the Six Sigma approach, joint application design for gathering requirements, usage of formal design methods, usage of structured coding techniques, and usage of libraries of proven reusable material.

The phrase *defect removal* includes all activities that can find errors or defects in any kind of deliverable. Examples of defect removal activities include requirements inspections, design inspections, document inspections, code inspections, and all kinds of testing.

Some activities benefit both defect prevention and defect removal simultaneously. For example, participation in design and code inspections is very effective in terms of defect removal, and also benefits defect prevention. Defect prevention is aided because inspection participants learn to avoid the kinds of errors that inspections detect.

As stated earlier, a combination of defect prevention and defect removal activities leads to some very significant differences in the overall numbers of software defects, compared between successful and unsuccessful projects [1]. However, additional data now shows that for projects in the 10,000 function point range the successful ones accumulate development totals of around 4.0 defects per function point and remove about 95 percent of them before delivery to customers. In other words, the number of delivered defects is about 0.2 defects per function point or 2,000 total latent defects. Of these, about 10 percent or 200 would be fairly serious defects. The rest would be minor or cosmetic defects.

By contrast, the unsuccessful projects accumulate development totals of around 7.0 defects per function point and remove only about 80 percent of them before delivery. The number of delivered defects is about 1.4 defects per function point or 14,000 total latent defects. Of these, 20 percent (or 2,800) would be fairly serious defects. This large number of latent defects after delivery is very troubling for users. The large number of delivered defects is also a frequent cause of litigation.

Unsuccessful projects typically omit design and code inspections and depend purely on testing. The omission of up-front inspections causes three serious problems: 1) The large number of defects still present when testing begins slows the project to a standstill; 2) The *bad fix* injection rate for projects without inspections is alarmingly high; and 3) The overall defect removal efficiency associated with only testing is not sufficient to achieve defect removal rates higher than about 80 percent.

Software Milestone Tracking

Those readers who work for the Department of Defense or for a defense contractor will note that the *earned value* approach is only cited in passing. There are several reasons for this. First, none of the lawsuits where the author was an expert witness involved defense projects so the earned-value method was not utilized. Second, although the earned-value method is common in the defense community, its usage among civilian projects including outsourced projects is very rare. Third, empirical data on the effectiveness of the earned-value approach is sparse. A number of defense projects that used earned-value methods have run late and been over budget. There are features of the earned-value method that would seem to improve both project estimating and project tracking, but empirical results are sparse.

Once a software project is under way, there are no fixed and reliable guidelines for judging its rate of progress. The civilian software industry has long utilized ad hoc milestones such as completion of design or completion of coding. However, these milestones are notoriously unreliable.

Tracking software projects requires dealing with two separate issues: 1) achieving specific and tangible milestones, and 2) expending resources and funds within specific budgeted amounts.

Because software milestones and costs are affected by requirements changes and *scope creep*, it is important to measure the increase in size of requirements changes, when they affect function point totals. However, there are also requirements changes that do not affect function point totals which are termed *requirements churn*. Both creep and churn occur at random intervals. Churn is harder to measure than creep and is often measured via *backfiring* or mathematical conversion between source code statements and function point metrics.

For an industry now more than 50 years old, it is somewhat surprising that there is not a general or universal set of project milestones for indicating tangible progress. From the author's assessment and baseline studies, Table 1 (see next page) shows some representative milestones that have shown practical value.

The most important aspect of Table 1 is that every milestone is based on completing a review, inspection, or test. Just finishing up a document or writing code should not be considered a milestone unless the deliverables have been reviewed, inspected, or tested.

Suggested Format for Monthly Status Reports for Software Projects

A suggested format for monthly progress tracking reports delivered to clients and higher management would include the following:

1. Status of last month's *red flag* problems.
2. New *red flag* problems noted this month.
3. Change requests processed this month versus change requests predicted.
4. Change requests predicted for next month.
5. Size in function points for this month's change requests.
6. Size in function points predicted for next month's change requests.
7. Schedule impacts of this month's change requests.
8. Cost impacts of this month's change requests.
9. Quality impacts of this month's change requests.
10. Defects found this month versus defects predicted.
11. Defects predicted for next month.
12. Costs expended this month versus costs predicted.
13. Costs predicted for next month.
14. Deliverables completed this month versus deliverables predicted.
15. Deliverables predicted for next month.

An interesting question is the frequency with which milestone progress should be reported. The most common reporting frequency is monthly, although exception reports can be filed at any time it is suspected that something has occurred that can cause perturbations. For example, serious illness of key project personnel or resignation of key personnel might very well affect project milestone completions – this kind of situation cannot be anticipated.

It might be thought that monthly reports are too far apart for small projects that last six months or less in total. For

1.	Requirements document completed.
2.	Requirements document review completed.
3.	Initial cost estimate completed.
4.	Initial cost estimate review completed.
5.	Development plan completed.
6.	Development plan review completed.
7.	Cost tracking system initialized.
8.	Defect tracking system initialized.
9.	Prototype completed.
10.	Prototype review completed.
11.	Complexity analysis of base system (for enhancement projects).
12.	Code restructuring of base system (for enhancement projects).
13.	Functional specification completed.
14.	Functional specification review completed.
15.	Data specification completed.
16.	Data specification review completed.
17.	Logic specification completed.
18.	Logic specification review completed.
19.	Quality control plan completed.
20.	Quality control plan review completed.
21.	Change control plan completed.
22.	Change control plan review completed.
23.	User information plan completed.
24.	User information plan review completed.
25.	Code for specific modules completed.
26.	Code inspection for specific modules completed.
27.	Code for specific modules unit tested.
28.	Test plan completed.
29.	Test plan review completed.
30.	Test cases for specific test stage completed.
31.	Test case inspection for specific test stage completed.
32.	Test stage completed.
33.	Test stage review completed.
34.	Integration for specific build completed.
35.	Integration review for specific build completed.
36.	User information completed.
37.	User information review completed.
38.	Quality assurance sign off completed.
39.	Delivery to beta test clients completed.
40.	Delivery to clients completed.

Table 1: *Representative Tracking Milestones for Large Software Projects*

small projects, weekly reports might be preferred. However, small projects usually do not get into serious trouble with cost and schedule overruns, whereas large projects almost always get in trouble with cost and schedule overruns. This article concentrates on the issues associated with large projects. In the litigation where the author has been an expert witness, every project in litigation except one was larger than 10,000 function points in size.

Failing or delayed projects usually lack serious milestone tracking. Activities are often reported as finished while work was still ongoing. Milestones on failing projects are usually dates on a calendar rather than completion and review of actual deliverables.

Delivering documents or code segments that are incomplete, contain errors, and cannot support downstream development work is not the way milestones are used by industry leaders.

Because milestone tracking occurs throughout software development, it is the last line of defense against project failures and delays. Milestones should be established formally and should be based on reviews, inspections, and tests of deliverables. Milestones should not be the dates that deliverables more or less were finished; they should reflect the dates that finished deliverables were validated by means of inspections, testing, and quality assurance review.

Summary and Results

Overcoming the risks shown here is largely a matter of opposites, or doing the reverse of what the risk indicates. Thus a well-formed software project will create accurate estimates derived from empirical data and supported by automated tools for handling the critical path issues. Such estimates will be based on the actual capabilities of the development team and will not be arbitrary creations derived without any rigor. The plans will specifically address the critical issues of change requests and quality control. In addition, monthly progress reports will also deal with these critical issues. Accurate progress reports are the last line of defense against failures.◆

References

1. Jones, Capers. "Social and Technical Reasons for Software Project Failure." CROSSTALK June 2006: 4-9.
2. Gibbs, T. Wayt. "Trends in Computing: Software's Chronic Crisis." Scientific American Magazine Sept. 1994: 72-81.

Note

1. This data is available in both CD and paper form <www.isbsg.org>.

About the Author

Capers Jones is currently the chairman of Capers Jones and Associates, LLC. He is also the founder and former chairman of Software Productivity Research (SPR) where he holds the title of Chief Scientist Emeritus. He is a well-known author and international public speaker, and has authored the books "Patterns of Software Systems Failure and Success," "Applied Software Measurement," "Software Quality: Analysis and Guidelines for Success," "Software Cost Estimation," and "Software Assessments, Benchmarks, and Best Practices." Jones and his colleagues from SPR have collected historical data from more than 600 corporations and more than 30 government organizations. This historical data is a key resource for judging the effectiveness of software process improvement methods.

Software Productivity Research, LLC
Phone: (877) 570-5459
Fax: (781) 273-5176
E-mail: capers.jones@spr.com, info@spr.com

Does Project Performance Stability Exist? A Re-examination of CPI and Evaluation of SPI(t) Stability

Kym Henderson
PMI College of Performance Management

Dr. Ofer Zwikael
Victoria University of Wellington

The development of the Earned Schedule (ES) method by Walt Lipke in 2003 has been shown to be an important extension to the Earned Value Management (EVM) method, increasing the utility of EVM data for project schedule analysis, control, and oversight. As ES provides a reliable time-based indicator of schedule performance, the objective of this article is to investigate whether the Schedule Performance Index (time) (SPI[t]) exhibited similar stability characteristics to those extensively reported for the Cost Performance Index (CPI)[1] in EVM. This article analyzes EVM data from three different countries for projects in three industry segments. There were 37 projects examined for SPI(t) stability and 26 for CPI stability. It has been found that while the behavior of SPI(t) is broadly consistent with CPI, the widely reported CPI stability rule cannot be generalized even within the U.S. Department of Defense (DoD) project portfolio. Further research is required to develop improved understanding of project performance characteristics and the behavior of CPI and the SPI(t).

The cancellation of the U.S. Navy's A-12 Avenger II stealth aircraft program in January 1991 resulted in research during the 1990s, which investigated the reliability of EVM cost prediction and the behavior of the CPI using DoD project[2] data [1, 2]. These research findings have come to be regarded as generally applicable across all project types using EVM across multiple industry sectors. A finding regarded as particularly significant was that CPI stabilizes by 20 percent of project completion.

Lipke proposed the ES method in 2003 to provide time-based measures of schedule performance utilizing EVM data. Initial validation has shown that the time-based ES-derived SPI(t) to be reliable for both early and late finish projects. For a technical description of the ES method, the reader is referred to [3]. For an excellent, easy-to-read, non-technical but comprehensive discussion of the ES method, refer to [4].

Following the initial validation of ES, interest developed in ascertaining whether SPI(t) exhibited similar stability characteristics to those extensively reported for CPI. The objective of this article is to re-examine CPI stability and to compare the stability behavior of the SPI(t) with CPI.

This article found that while the behavior of the SPI(t) is broadly consistent with CPI, the widely reported CPI stability rule cannot be generalized to all projects utilizing the EVM method or even within the DoD project portfolio. However, the consistent behavior to CPI demonstrated by SPI(t) provides further support for the validity of the SPI(t) metric and the ES method.

Additional analysis was unable to establish a correlation between achieving earlier CPI and SPI(t) stability and improved outcomes at completion. In certain cases, where projects achieved either under budget and/or early finish outcomes with cost and/or schedule stability achieved late, earlier cost and/or schedule stability would have been disadvantageous to the actual final outcome(s) achieved. This is because CPI and/or SPI(t) progressively improved over the life of those projects.

This article also demonstrates that by utilizing ES, research of schedule performance using EVM data is now possible and leads to improved understanding of the dynamics of project schedule and project cost performance.

Background

The CPI has long been a key indicator used to analyze the cost performance of projects using EVM. The first empiric confirmation of the widely reported and referenced CPI stability rule was by Christensen and Payne, using data from 26 U.S. Air Force-completed contracts in 1992. The data used came from the cost library of the U.S. Air Force Systems Command Aeronautical Systems Division [5].

Christensen and Templin conveniently summarized the series of research findings subsequent to [5] in 2002:

> … the range of the cumulative CPI from the 20 percent completion point to contract completion was less than 0.20 for every contract. This result is usually interpreted to mean that the cumulative CPI does not change by more than plus or minus 0.10 from its value at the 20 percent completion point, and is used to evaluate the reasonableness of projected cost efficiencies on future work [6].

Christensen and Payne made the following observations on the perceived importance of CPI stability:

- A stable CPI is evidence that the contractor's management control systems, particularly the planning, budgeting, and accounting systems, are functioning properly.
- A stable CPI may thus indicate that the contractor's estimated final costs of the authorized work, termed *Estimated at Completion* (EAC), are reliable.
- In addition, knowing that the CPI is stable may help the analyst evaluate the capability of a contractor to recover from a cost overrun by comparing the CPI with other key indicators, such as the To-Complete Performance Index [5].

Over time, the widely reported CPI stability findings have been generalized as being applicable to all projects utilizing the EVM method [7-11]. An extensive literature review has not found further empiric validation of the CPI stability rule beyond the project data obtained in the initial paper and data from the DoD Defense Acquisition Executive Summary (DAES) database.

Concurrent research into the stability characteristics of the EVM SPI was not possible because the SPI is known to fail as a statistical predictor because it always returns to unity at project completion, irrespective of duration-based delay. The SPI is also recognized as failing nominally within the final third of the project, and it also fails after the project's planned duration has been exceeded.

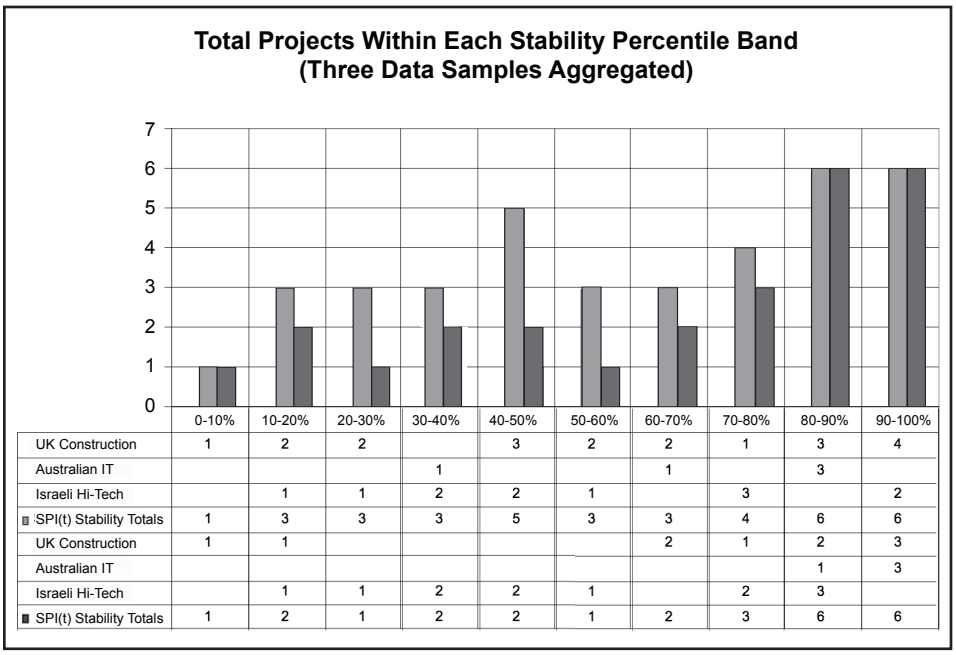
Lipke proposed the ES method in 2003 as a solution to these limitations and flaws of the EVM schedule indicators [3]. A series of studies provided initial valida-

	CPI Stability		SPI(t) Stability	
	Test Statistic	Test Result	Test Statistic	Test Result
UK Construction	0.623	Ho	0.748	Ho
Australian IT	1.000	Ho	0.500	Ho
Israeli Hi-Tech	0.806	Ho	0.613	Ho
Composite	0.916	Ho	0.629	Ho

Table 1: *Hypothesis Test Results*

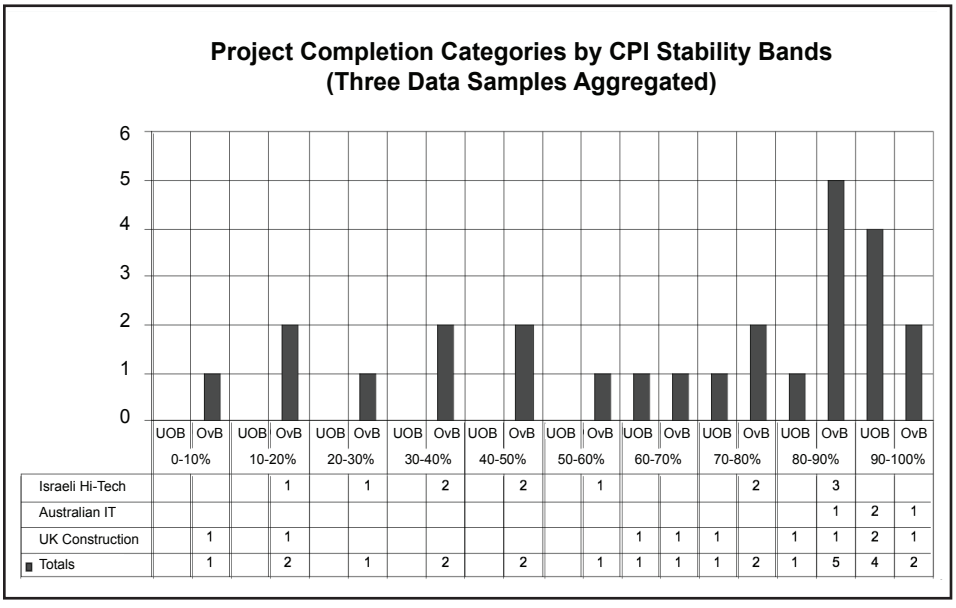
Stability Achieved		UK Construction	Australian IT	Israeli Hi-Tech	Composite
SPI(t) cum.	**≤ 20%**	3	0	1	4
> 20%		17	5	11	33
CPI cum.	**≤ 20%**	2	0	1	3
> 20%		8	4	11	23

Table 2: *Summary of Stability Achievement Related to 20 Percent Completion*



	0-10%	10-20%	20-30%	30-40%	40-50%	50-60%	60-70%	70-80%	80-90%	90-100%
UK Construction	1	2	2		3	2	2	1	3	4
Australian IT				1			1		3	
Israeli Hi-Tech		1	1	2	2	1		3		2
SPI(t) Stability Totals	1	3	3	3	5	3	3	4	6	6
UK Construction	1	1					2	1	2	3
Australian IT									1	3
Israeli Hi-Tech		1	1	2	2	1		2	3	
SPI(t) Stability Totals	1	2	1	2	2	1	2	3	6	6

Figure 1: *Total Projects CPI and SPI(t) Stability Within Each 10 Percentile Band*



	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB	UOB	OvB
	0-10%		10-20%		20-30%		30-40%		40-50%		50-60%		60-70%		70-80%		80-90%		90-100%	
Israeli Hi-Tech				1		1		2		2		1				2		3		
Australian IT																		1	2	1
UK Construction		1		1									1	1	1		1	1	2	1
Totals		1		2		1		2		2		1	1	1	1	2	1	5	4	2

Figure 2: *Project Completion Categories by CPI Stability Band*

tion of the ES method, some by using real EVM project data and also by using simulated work schedules [12-16]. The time-based ES derived SPI(t) has been shown to be reliable for both early and late finish projects. The SPI(t) only reverts to unity at project completion if on-time completion has been achieved.

A research study intended to validate the ES construct using DAES data was commissioned in 2004 and undertaken by a U.S. Air Force Institute of Technology masters student. Unfortunately, this study was discontinued after an independent review determined the following:

> Results: The historical data collection procedures for the DoD and U.S. Air Force do not allow for sufficient testing of ES theory at this time. A statistical evaluation concluded that SPI(t) is different than SPI($); however, the two variables are highly correlated. The result of the analysis identified that SPI(t) performs similarly to SPI($) with the data contained in the DAES database. In order for the ES theory to be fully investigated, additional data must be collected. This research shows that the necessary data may also not be available despite the best collection efforts. The original schedule and planned duration information is critical to successful evaluation of the ES methodology [17].

However, early interest by the Project Management Institute (PMI) resulted in the principles of ES being included as an *Emerging Practice Insert* in the Practice Standard for Earned Value Management published in 2004 [18].

Following the initial validation of ES, interest developed in ascertaining whether the SPI(t) exhibited similar stability characteristics to those extensively reported for the CPI. The objective of this article is to re-examine CPI stability and to compare the stability behavior of the SPI(t) with CPI.

Method for Evaluating Stability

EVM project data was loaded into a Microsoft Excel Stability Point Calculator[3] developed by Lipke. The calculator determines the observation number in a sequence of CPI and SPI(t) values at which all subsequent observations are within a defined stability limit. The stability limit used is 0.10. The calculator enables

the associated percentage complete at which stability occurs to be determined.

To determine the significance of the observations of stability for both CPI and SPI(t), statistical hypothesis testing is conducted. The test applied is the Sign Test at 0.05 level of significance[4] [19]. The Sign Test was used in this research because it does not depend upon the data having a normal distribution. In past research, the hypothesis test method chosen implied that the data was normally distributed; however, the normality of the data was not established. Research by Lipke also suggests the following:

> Results indicate the logarithm data representations of the indexes are likely normally distributed, whereas the distributions for CPI, SPI, and CV are not [20].

The question to answer regarding stability is *can it be stated generally and reliably that the final value of the performance index is within 0.10 of its value when the project is 20 percent complete?* The answer to the question will be yes if the alternate hypothesis is satisfied:

H1(CPI): |CPI(final) – CPI(20%)| < 0.10

H2 (SPI(t)): |SPI(t)(final) – SPI(t)(20%)| < 0.10

Two separate hypothesis tests are conducted, one for CPI and one for the SPI(t). The result from the hypothesis testing is recorded as Ha when the value of the test statistic is in the critical region (0.05) and Ho (null hypothesis) when it is not.

The Data

A composite EVM data set was assembled comprising commercial sector data samples obtained from following:

- Twenty-four United Kingdom (UK) construction projects.
- Twelve Israeli high-technology (Hi-Tech) projects.
- Nine Australian Information Technology (IT) projects.

The EVM data consists of direct labor costs only with the following:

- UK construction projects recorded in *person days* weekly with EVM values expressed as a percentage of the budget at complete to further maintain data anonymity.
- Israeli Hi-Tech projects recorded in U.S. dollars monthly.
- Australian IT projects recorded in Australian dollars weekly.

An extensive review of the data was undertaken. Projects were excluded from the sample for a variety of reasons including the following:

- Lack of data integrity.
- Lack of EV data at 20 percent of project completion.
- Partially incomplete Planned Value data.
- Lack of required Actual Cost (AC) data.

Ten UK construction projects are included in the CPI stability research sample. Five of these project were included although the final AC data available was between 96.7 percent and 99.0 percent complete. Including those five projects is consistent with the approach adopted by Christensen and Payne's research and assumes that the difference between CPI_{Final} and the latest available CPI has no material impact on the findings [5].

The outcome was a usable data sample of the following:

- Twelve Israeli Hi-Tech projects for the SPI(t) and CPI stability research.
- Twenty UK construction projects for the SPI(t) stability and 10 for CPI stability research.
- Five Australian IT projects for the SPI(t) stability and four for CPI stability research.

Stability Evaluation Results

The results of the sign tests for the following hypothesis are shown in Table 1: Can it be stated generally and reliably that the final value of the performance index is within 0.10 of its value when the project is 20 percent complete? Recall that the test result of Ha indicates stability of the performance indicators CPI and the SPI(t). As is shown, the test results did not have any test statistic in the critical region (0.05). As a result, none of the null hypotheses can be rejected for any of the three samples or the composite of all samples. This means that stability was not achieved for either CPI or the SPI(t) by the time the project was 20 percent complete.

This research does not support the previously referenced generalizations that the CPI stability rule has universal applicability for all projects utilizing the EVM method. Because the SPI(t) index demonstrates a similar lack of stability to that found for CPI, the validity of the SPI(t) metric is supported due to the consistent behavior demonstrated with CPI.

Table 2 summarizes the raw data in relation to the numbers of projects that achieved stability before or after 20 percent completion for the SPI(t) and CPI by each project set and for the composite of all. It can be seen that the majority of projects reach stability only after the 20 percent completion point.

Figure 1 summarizes each 10 percent complete percentile band where CPI and the SPI(t) stability occurred. This figure shows the following:

- The wide variability in the achievement of stability for both CPI and the SPI(t). Project performance heuristics or *rules of thumb* intended to be generally applicable (e.g., the CPI stability rule) require an empirically established consistency of behavior across a broad range of projects. These findings are a significant impediment to proposing and confirming broadly applicable CPI and SPI(t) stability heuristics.
- That stability is usually achieved very late in the project life cycle, often later than 80 percent complete for projects in these samples.

Zwikael analyzed the Israeli Hi-Tech project sample using visual inspection of charts and suggested that CPI stability was, on average, achieved at the 60 percent completion point [21]. That analysis broadly confirms this article's finding of CPI stability being achieved much later in the project life cycle than previously reported.

Additional Analysis

Following the lack of CPI and SPI(t) stability findings additional analysis was conducted. Within each 10 percent complete percentile band projects were categorized as follows:

- Cost at completion:
  - Under or On Budget (UOB).
  - Over Budget (OvB).
- Schedule at completion:
  - Early or On Time finish (EOT).
  - Late Finish (LF).

The purpose of this analysis is to determine if there is a correlation between achieving earlier CPI and the SPI(t) stability and improved project outcomes.

Figure 2 summarizes the analysis for CPI and Figure 3 (see next page) does the same for the SPI(t). With the data samples utilized, achievement of earlier stability is not correlated with improved final cost and/or schedule outcomes.

For UOB and EOT projects where cost and schedule stability was achieved late (after, say, 60 percent completion) achieving earlier stability would have been disadvantageous to the final outcome(s) achieved because project performance progressively improved over the life of those projects.

Figure 4 summarizes projects (with the required comparative data), which

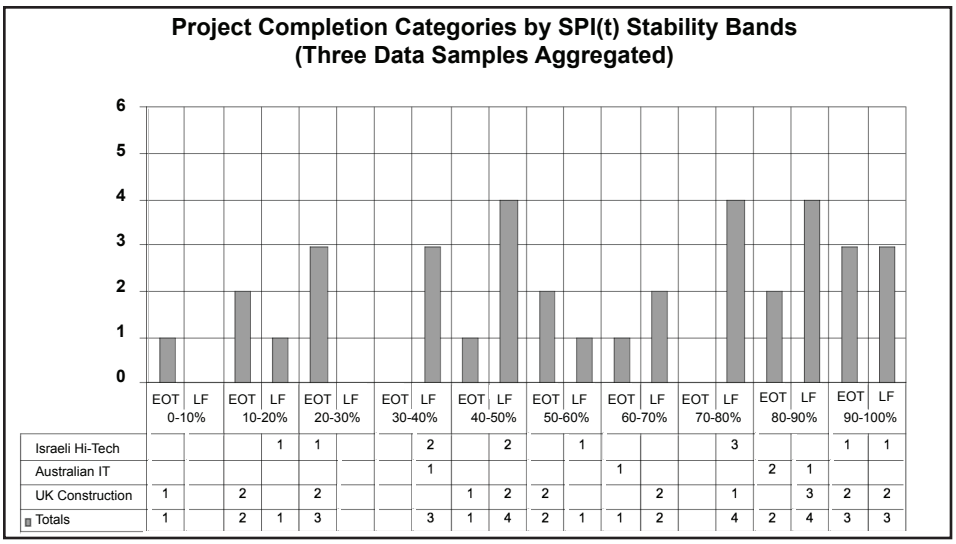


Figure 3: *Project Completion Categories by SPI(t) Stability Band*

achieved SPI(t) or CPI stability first. Achieving SPI(t) stability first implies schedule management had a higher management priority; achieving CPI stability first implies cost management had the higher priority.

In the Australian IT projects sample, SPI(t) stability was achieved first for the preponderance of projects. For the other data samples, the achievement of cost or schedule, stability first occurred in roughly equal proportion. In only one project in these samples – an Australian IT project – was the cost and schedule stability achieved simultaneously.

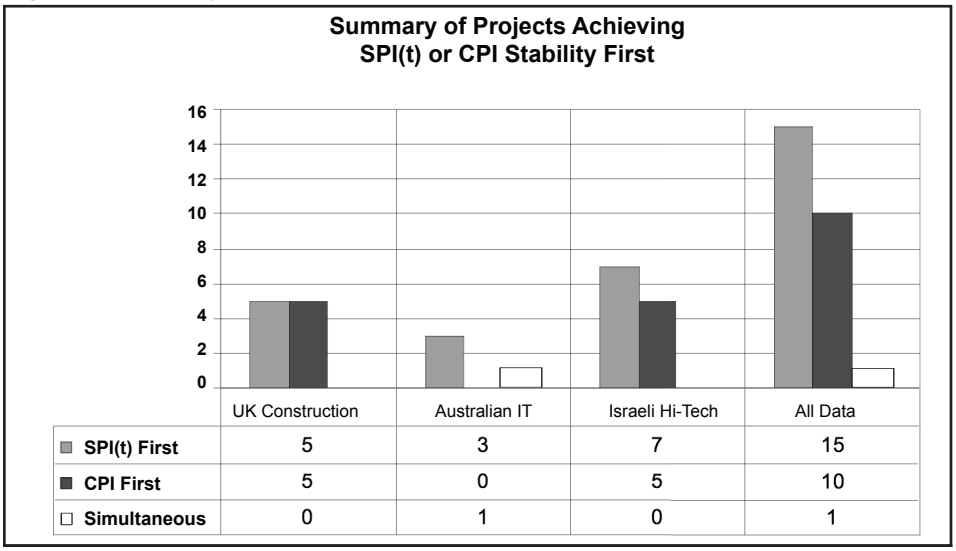
Corroboration With Other Research

Because of the comprehensive contradiction to the previously published CPI stability research findings, a further literature review was undertaken. This review obtained a most unexpected source of independent corroboration for this article's CPI stability findings. In the mid-90s, Michael Popp, a civilian employee of the U.S. Naval Air Command (NAVAIR), initiated an internal DoD research project within NAVAIR.

The output was an internal but unclassified NAVAIR report (the Popp report) which has, with Popp's permission, now been placed into the public domain on the PMI Sydney Chapter Web site [22]. The purpose of the Popp study was to develop probability distributions of cost EACs based on the CPI at complete, current CPI, and percentage complete of projects based on history. As stated in the report: Given a program has a CPI of X and a percent complete of Y, what is the most likely finishing CPI [22]?

In contrast to Christensen and associates research, which used data from the DAES database, the data used by Popp was sourced from the Contracts Analysis System database maintained by the Office of the Secretary of Defense Cost Analysis Improvement Group.

Figure 4: *Summary of Projects Achieving SPI(t) or CPI Stability First*



The research undertaken by Popp did not focus on CPI stability. However, charts which can also be used for assessing CPI stability were completed as part of that study. These charts correlate the cumulative CPI for the percentage complete in each 10 percent complete percentile band to the CPI_{Final} for all projects in that sample.

Figure 5 is the first chart of interest from the Popp report, as it shows the correlation between the cumulative CPI at 10-20 percent complete and the CPI Final for all projects in the sample.

The area of the chart enclosed within the dashed lines bounds the area in which the correlation plots must occur for the Christensen derived CPI stability rule to apply. Those plots which occur outside the enclosed area are also in conflict with the Christensen derived CPI stability rule. The limited data samples used in this analysis are sufficient to show that the CPI stability rule cannot be generalized even within the DoD project portfolio.

While research using the Popp report data sample was not principally directed at examining the validity of the CPI stability rule, this research found the following:

- Development programs *at 20 percent (completion), programs with a cumulative CPI below 0.89 improve* which was *close to Christensen (findings), but with some exceptions*.
- Production programs *at 20 percent (completion), programs with a cumulative CPI below 0.84 improve*, again *close to Christensen (findings), but with some exceptions* [23].

Using the *enclosure* technique, Figure 6 shows that the preponderance of plots occur within the area where the CPI stability rule applies at 20 percent completion. The conclusion is that for the DoD project data used by Popp, CPI stability was also achieved very late in the project life cycle, often as late as 70-80 percent completion. This finding is consistent with the late CPI stability findings for the commercial sector project samples as shown in Figure 1.

While the underlying data was not available and further research is required, these findings also conflict with the DoD research findings quoted in the Beach report into the A-12 cancellation:

> DoD experience in more than 400 programs since 1977 indicates without exception that the cumulative CPI does not significantly improve during the period between 15% and 85% of contract

performance; in fact, it tends to decline [1].

Some projects in the Popp sample show a trend of CPI performance improvement, from $CPI_{20\ percent}$ and in a smaller number of cases, as late as $CPI_{80\ percent}$ to CPI_{Final}.

Summary and Conclusion

The initial objective of this article – ascertaining whether the SPI(t) demonstrates similar stability characteristics to those extensively reported for CPI – was not achieved. This article has found that while the behavior of the SPI(t) is broadly consistent with CPI, the widely reported CPI stability rule cannot be generalized to all projects using the EVM method or even within the DoD project portfolio. However, the consistent behavior to CPI demonstrated by the SPI(t) provides further support for the validity of the SPI(t) metric and the ES method.

Additional analysis was unable to establish a correlation between achieving earlier CPI and the SPI(t) stability and improved outcomes at completion. In cases where projects achieved either under budget and/or early finish outcomes with cost and/or schedule stability achieved late (i.e., after, say, 60 percent completion), earlier cost and/or schedule stability would have been disadvantageous to the actual final outcome(s) achieved. This is because CPI and/or the SPI(t) were progressively improving over the life of those projects.

The findings and corroboration of this article require significant review and revision to what has been regarded as a long settled EVM heuristic with regard to CPI stability and consequent practice including the use of a stable CPI as evidence that an EVM system is functioning properly and of a *reliable* EAC [5].

Improvements to current EVM techniques for predicting future cost performance should be considered as current techniques have relied on generalizing research findings from limited data sources, principally the DAES database.

Alternatives methods of cost and schedule prediction using well-established statistical principles and methods developed by Lipke show the following promise:

- These techniques allow generation of a range of cost and schedule predictions from user defined Confidence Limit(s).
- All information and data required for these predictions comes from within the project itself.

This may reduce the current dependence on heuristics developed from external project data sources, which might not be applicable to the project of interest.

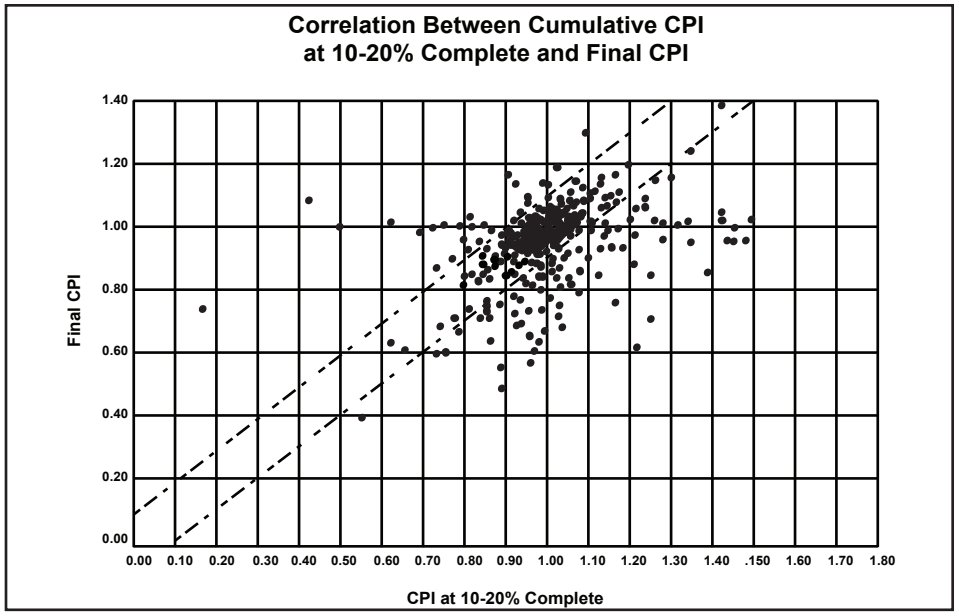


Figure 5: *Correlation Between Cumulative CPI at 10-20 Percent Complete and Final CPI (Popp)*

To promote trials of these statistical prediction techniques, a freely available calculator can be found on the ES Web site[3]. An academic article fully describing the statistical prediction techniques and supporting rationales is pending publication [24]. The statistical prediction techniques developed have been summarized in a presentation by Henderson which is available at [25].

A major advance to EVM practice and future research opportunities would be development of a broadly based EVM research database where completed EVM project data could be submitted anonymously for the following:

- Researching purposes.
- Benchmarking completed project performance.
- Assisting in the sizing of projects.

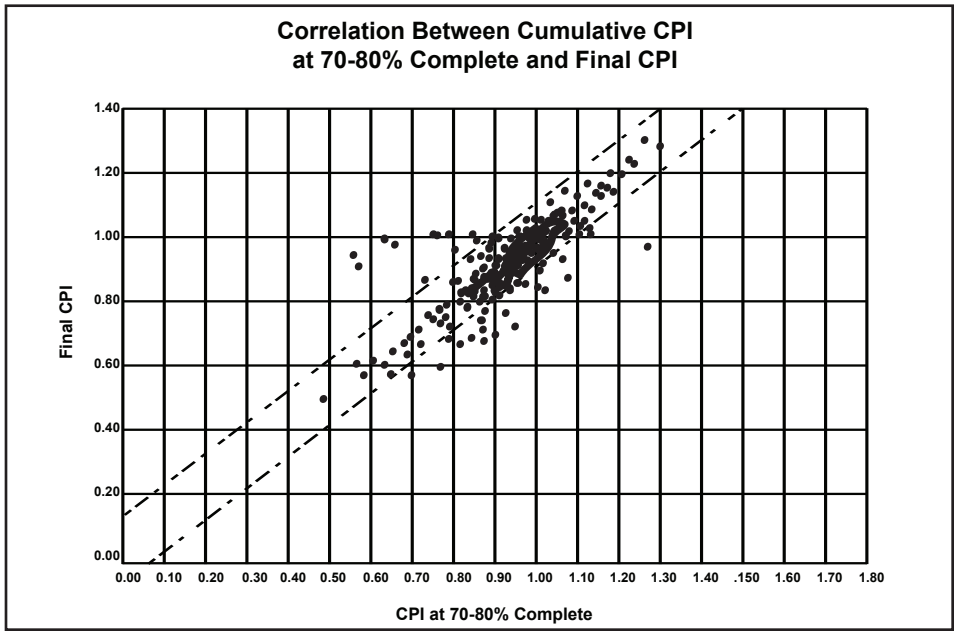
Such knowledge bases are not unique in other disciplines, with an instructive Australian example being the ISBSG Web site at <www.isbsg.org>.

Improved data collection techniques to ensure that baseline schedule information is captured and stored in the DAES database are also recommended.

Final Remarks and Future Research

While this article has overturned long-standing findings and beliefs on CPI stability, it is important that the strengths and

Figure 6: *Correlation Between Cumulative CPI at 70-80 Percent Complete and Final CPI (Popp)*



Coming Events

April 29-May 2

2008 Systems and Software Technology Conference
Las Vegas, NV
www.sstc-online.org

May 5-9

PSQT West 2008
International Conference on Practical Software Quality and Testing
Las Vegas, NV
www.psqtconference.com

May 5-9

STAR EAST 2008
Software Testing Analysis and Review
Orlando, FL
www.sqe.com/StarEast

May 6

SLAAD 2008
Strike, Land Attack and Air Defense Division Annual Symposium
Laurel, MD
www.ndia.org

May 13-15

Continuous Process Improvement Symposium
Ogden, UT
www.cpi-symposiums.com/info.html

May 14-15

8th Annual ISSEA Conference
International Systems Security Engineering Association
Chicago, IL
www.issea.org

Coming Events: Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: nicole.kentta@hill.af.mil.

limitations of the EVM method are properly understood, particularly in the following areas:

- Adoption of EVM by U.S. government agencies through the Office of Management Budget Circular A-11 Part 7 mandate.
- Advocacy of the use of EVM cost predictors to assess compliance to the Sarbanes-Oxley Act [9].
- Increased interest and the adoption of EVM by organizations globally.

Where projects have not exhibited CPI *stability*, EVM practitioners can now know that this is neither unique, nor is it necessarily an adverse reflection on the management or execution of those projects.

Various follow-on research opportunities arise from this article, which may develop improved understanding of project performance characteristics and generalizable heuristics. Suggestions include examining the performance characteristics of projects where the following happens:

- The CPI stability rule does seem applicable (e.g., the subset highlighted in the Popp report data) to determine whether there are project characteristics which result in early CPI stability.
- Early CPI stability was not achieved due to progressively improving CPI performance over the project life cycle.

Academically oriented research aimed at establishing a theoretical rationale for project performance instability would be another useful addition to the project management body of knowledge.

While [23] provides the sobering assessment consistent with Christensen's findings *average to good programs do not improve*, an understanding of project characteristics, which result in progressively improving CPI would, if these characteristics could be emulated in other programs, be an extremely useful advance to practice. Such research could offer significant opportunities for tangibly improving project performance.

Research opportunities are equally applicable to project schedule performance. This article also demonstrates that by using ES, research of schedule performance using EVM data is possible and already leading to improved understanding of the dynamics of project schedule and project cost performance.◆

Acknowledgements

This research has been made possible due to the generous assistance of the following individuals:

- The project controls manager from the UK-based construction company (who desire anonymity) for making available the UK construction projects' EVM data.
- Michael Popp of NAVAIR for making available the Popp Report and providing permission for the report to be placed in the public domain on the PMI Sydney Chapter Web site [22].

The support, suggestions, general assistance, and review comments by the ES advocates and researchers, which significantly improved this article is also appreciated. Responsibility for any errors, omissions or erroneous conclusions remains the sole responsibility of the authors.

References

1. Beach, Chester Paul, Jr. "A-12 Administrative Inquiry. Report to the Secretary of Navy." Washington D.C.: Department of the Navy, 1990 <www.suu.edu/faculty/christensend/evms/beacha-1.pdf>.
2. Christensen, David S. "Using the Earned Value Cost Management Report To Evaluate The Contractor's Estimate at Completion." Acquisition Review Quarterly Summer 1999: 283-295 <www.dau.mil/pubs/arq/99arq/chrisevm.pdf>.
3. Lipke, Walt. "Schedule Is Different." The Measurable News Mar. 2003: 10-15 <www.earnedschedule.com/Docs/Schedule%20is%20Different.pdf>.
4. Stratton, Ray. "Not Your Father's Earned Value." Projects at Work <www.projectsatwork.com>, <www.earnedschedule.com/Docs/Not%20Your%20Father%27s%20Earned%20Value.pdf>.
5. Christensen, David S., and Kirk Payne. "Cost Performance Stability – Fact or Fiction?" Journal of Parametrics 10 (1992): 27-40 <www.suu.edu/faculty/christensend/evms/CPIstabilityJP.pdf>.
6. Christensen, David S., and Carl Templin. "EAC Evaluation Methods: Do They Still Work?" Acquisition Review Quarterly (2002): 105-116 <www.suu.edu/faculty/christensend/evms/eacevalmethods4.pdf>.
7. Christensen, D.S., and S.R. Heise. "Cost Performance Index Stability." National Contract Management Journal 25 (1993): 7-15 <www.suu.edu/faculty/christensend/evms/CPIstabilityNCMJ.pdf>.
8. Fleming, Quentin, and Joel Koppelmann. The Earned Value Body of Knowledge. Proc. of the 30th Annual PMI 1999 Seminars and Symposium, Philadelphia, PA.
9. Fleming, Quentin, and Joel

Koppelmann. "Sarbanes-Oxley: Does Compliance Require Earned Value Management on Projects." Contracts Management Apr. 2004: 26-28.
10. Fleming, Quentin, and Joel Koppelmann. "If EVM Is Good … Why Isn't It Used On All Projects?" Contracts Management Apr. 2004: 26-28 <www.suu.edu/faculty/christensend/evms/WhyEVM.pdf>.
11. Fleming, Quentin, and Joel Koppelman. Earned Value Project Management. 3rd ed. Upper Darby, PA: Project Management Institute, 2005.
12. Henderson, Kym. "Earned Schedule: A Breakthrough Extension to Earned Value Theory? A Retrospective Analysis of Real Project Data." The Measurable News Summer 2003: 13-23 <www.earnedschedule.com/Docs/Earned%20Schedule%20-%20A%20Breakthrough%20Extension%20to%20EVM.pdf>.
13. Henderson, Kym. "Further Developments in Earned Schedule." The Measurable News Spring 2004: 15-22 <www.earnedschedule.com/Docs/Further%20Developments%20in%20Earned%20Schedule.pdf>.
14. Henderson, Kym. "Earned Schedule in Action." The Measurable News Spring 2005: 23-30 <www.earnedschedule.com/Docs/Earned%20Schedule%20in%20Action.pdf>.
15. Vanhoucke, Mario, and Stephan Vandevoorde. "A Comparison of Different Project Duration Forecasting Methods Using Earned Value Metrics." International Journal of Project Management 24.4 (2006): 289-302 <www.sciencedirect.com>.
16. Vanhoucke Mario, and Stephan Vandevoorde. "A Simulation and Evaluation of Earned Value Metrics to Forecast Project Duration." Journal of Operational Research Society 58.10 (2007):1361-1374 <www.palgrave-journals.com>.
17. Witte, Ed. "An Analysis of the Schedule Performance Index (SPI) in Units of Time: Overcoming the SPI($) Limitations to Accurately Portray Schedule." Personal e-mail to Walt Lipke. 15 Apr. 2005.
18. PMI. Practice Standard for Earned Value Management. PMI, 2004.
19. National Institute of Standards and Technology Dataplot. Sign Test. 2005 <www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/signtest.htm>.
20. Lipke, Walt. "A Study of the Normality of Earned Value Management Indicators." The Measurable News Dec. 2002: 1-16 <sydney.pmichapters-australia.org.au/programs/customer/v_filedown.asp?P=31&FID=89297847&FRF=n&>.
21. Zwikael, Ofer, et al. "Evaluation of Models for Forecasting the Final Cost of a Project." Project Management Journal 31.1 (2000): 53-57.
22. Popp, Michael. "Probability Distributions of CPI at Complete vs. CPI Today." Internal NAVAIR Report, Unpublished, 1996. <sydney.pmichapters-australia.org.au/programs/customer/v_filedown.asp?P=31&FID=738016087&FRF=n&>.
23. Coleman et al. "Predicting Final CPI." Presentation to the 4th Joint Annual ISPA/SCEA International Conference, Orlando, FL, June 2003.
24. Lipke, Walt, et al. "Prediction of Project Outcome – The Application of Statistical Methods to Earned Value Management and Earned Schedule Performance Indexes." Publication pending.
25. Henderson, Kym. "Recent Advances in Project Prediction Techniques." Presentation to the IQPC IT Project Management Conference, Sydney Australia, 1 May 2007 <www.earnedschedule.com/Docs/Recent%20Advances%20in%20Project%20Prediction%20Techniques.pdf>.

Notes

1. Unless otherwise stated, all references to CPI and the SPI(t) refer to the cumulative values.
2. Project has been used consistently throughout this article. In the U.S. government, particularly the DoD context, *program* may be the more appropriate term.
3. This calculator has been placed into the public domain to encourage more broadly based CPI and SPI(t) stability research and is freely available from the ES Web site at <www.earnedschedule.com/Calculator.shtml>.
4. Applying the Sign Test at 0.05 level of significance means that the test is being applied at a 95 percent level of confidence.

About the Authors

Kym Henderson is a practicing IT project manager with significant experience in project recoveries utilizing simplified EVM techniques. He has presented at many conferences internationally and published papers in various publications and as proceedings of PMI Global Congresses. Henderson published the first independent validation of the ES method in 2003. He is the Immediate Past Education Director (2003-2007) of the PMI Sydney Chapter and is the first non-U.S. national elected to the board of the PMI College of Performance Management commencing office on 1st January 2008. He has a bachelor of business and a master of science (computing) from the University of Technology, Sydney.

P.O. Box 687
Randwick NSW 2031
Australia
Phone: +61 414 428 537
Fax: +61 2 8394 9295
E-mail: kymhenderson @froggy.com.au

Ofer Zwikael, Ph.D., is a senior lecturer at the Victoria Management School, Victoria University of Wellington, New Zealand. He also leads projects and program groups in dozens of organizations, in Asia and Europe. Zwikael is an accredited Project Management Professional, has acted for two years as a vice president in the Executive Board of the PMI's Israeli chapter, and is currently a director at the New Zealand PMI Executive Board.

Victoria Management School
Victoria University of Wellington
Rutherford House
23 Lambton Quay
P.O. Box 600
Wellington
New Zealand
Phone: +64 4 4635143
E-mail: ofer.zwikael@vuw.ac.nz

Schedule Adherence: A Useful Measure for Project Management

Walt Lipke
PMI Oklahoma City Chapter

Earned Value Management (EVM) is a very good method of project management. However, EVM by itself cannot provide information as to how the schedule is being accomplished. Project accomplishment not in accordance with the planned schedule frequently has adverse repercussions; cost increases and duration is elongated. Thus, managers have a need to more fully understand project performance. This article utilizes the new practice of Earned Schedule (ES) to discuss a proposed measure for further enhancing the practice of EVM. The measure, Schedule Adherence, provides additional early warning information to project managers, thereby enabling improved decision making and enhancing the probability of project success.

Development of a plan for executing a project is a difficult undertaking. When the plan is being created, a work flow is envisioned along with constraints and resource availability. There is a considerable amount of effort invested in decomposing the constituents of the plan into manageable components and work packages. Detailed examination of the tasks themselves is made to prepare reasonable estimates for their cost and duration. Oftentimes, planning teams use historical project records, heuristics, and statistical algorithms to determine best and worst case probable outcomes. Furthermore, to assure that the best possible plan is created, technical experts may be employed to make the estimates as accurate as possible.

Before assignments can be made to the team members of a project, the timing of their actions must be known along with their interdependencies. The intricate mechanism for consolidating all of this information and making it understandable to the project team and senior management, as well, is the *schedule*. The schedule is an embodiment of our best understanding of how to accomplish the project ... a truly important document. Possibly, the schedule is the single most important document pertaining to the project, and it likely has more to do with success than any other aspect.

Well, then, if the planned schedule is so crucial to project success, it follows that project managers should do their utmost to ensure project execution conforms to it. Assuming the planned schedule is the most efficient path for executing the project, any deviation leads to inefficiency and very likely other problems such as constraint reduced production, idle time, skills mismatch, and poor quality output, and in turn, requires rework. Thus, there is an extremely compelling case for following the planned schedule.

® Capability Maturity Model is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

This article presents a proposed method for measuring the conformance, or adherence, for the schedule execution of a project. Utilizing the method and measure, the project manager has a better understanding of how well the execution follows the sequence and precedence of the tasks in the baseline schedule. Having an indicator for *schedule adherence* provides additional early warning information for managers to act upon.

Schedule Performance Efficiency Versus Schedule Adherence

What is meant by *schedule adherence*? Does it mean that the project is performing such that objectives are achieved at the time predicted or planned? Certainly project managers want to know that interim products are being produced and delivered on time. This type of schedule performance indicator can be made a number of different ways, such as portion or percent of milestones, objectives, or interim products achieved on time. In fact, the EVM Schedule Performance Indicator (SPI) is of this type[1]. However, SPI is much more resolute than the very coarse measures mentioned; its increment of measure is cost – earned and planned. This discussion for SPI is equally applicable to the time-based schedule performance efficiency indicator from ES, SPI(t)[2].

All of these indicators, including SPI and SPI(t), describe the efficiency of achieving the plan. However, they do not provide information about how the products, milestones, objectives, or earned value were achieved. For example, these indicators cannot describe whether or not completion of milestone 2 followed milestone 1. If the milestone schedule indicates that at status period 3 we should have completed two milestones and we have completed two, it would appear from the indicator (milestone percent completed = 100 percent), that all is well. But what if the two milestones are numbers one and three while the second milestone is still in work? Is there anything possibly wrong? After all, the project has met its two-milestone objective.

For the EVM schedule efficiency indicator, SPI, there is no concern as to whether the earned value (EV) accrued matches the expectation of the schedule. In most cases, project managers would celebrate an SPI = 1.0 because it is so seldom achieved, and consequently would not question whether the EV accrued is, in fact, the expected planned value (PV). Again, the question is raised: *Should the project manager be concerned with the performance sequence, i.e., how the achievement occurred*? Does it make any difference?

Over the last 20 years, nearly every industry experienced several initiatives intended to improve project performance and product quality: Statistical Process Control, Total Quality Management, the Software Engineering Institute Capability Maturity Model®, and the International Organization Standard for Quality Management Systems 9001. The fundamental idea from all of these process improvement efforts is the following: *Undisciplined execution leads to inefficient performance and defective products.*

Does this thinking apply to project plans, too? Of course it does; the planned schedule describes the execution process. Therefore, it is not enough to measure the execution efficiency. Additionally, project managers (PM) need to know how well the process is being followed. By maintaining process integrity, PMs can maximize the project's performance and minimize its rework and delivery of defective products. An indicator for adherence to the schedule provides the measure needed by PMs for monitoring and controlling the project execution.

Measuring and Indicating Schedule Adherence

The idea for measuring schedule adherence is simply stated in this question: *Did*

the accomplishment match exactly the expectation from the planned schedule? This is not the same as the preceding discussion of schedule performance efficiency, where the volume of actual work accomplished is compared to the expected volume from the schedule. Schedule adherence is a more restrictive measure, and it is independent from performance efficiency.

A recent enhancement to EVM, ES, provides a means to measure schedule adherence. ES is derived from two measures of EVM, PV, and EV [1]. The accumulated planned value from the project start to its planned completion is the performance measurement baseline (PMB) [2]. ES is the time duration associated with the PMB where the PV is equal to the EV accrued.

The concept of ES is illustrated by Figure 1. Arrow A projects the accrued value of EV onto the PMB to identify the point at which PV equals EV. Arrow B identifies the time at which PV equals the EV accrued, i.e., the planned duration earned or ES. The time at which the EV accrued appears is period seven. Whereas ES is determined to be the duration of five periods; i.e., the time measure from the PMB where PV is equal to the EV accrued at Time Now, or Actual Time (AT).

Two comparative measures, SV_t and SV_c, are shown in the diagram to illustrate the difference between the cost-based and time-based indicators of EVM and ES, respectively. The traditional EVM schedule variance is SV_c, while the time-based schedule variance from ES is SV_t[3]. From the numbers shown in the diagram, SV_t can be easily computed: $SV_t = ES - AT = 5\text{-}7 = -2$. Assuming the units are months, the project is two months behind its planned schedule.

The performance expectation for the planned schedule is embodied in the PMB. This is a consequence of the PMB being the result from summing time phased PV across all tasks in the schedule. Figure 2 is used to illustrate the relationship. The figure shows a network schedule at the top with the PV curve beneath it.

The connection between EVM and the schedule provided by ES is remarkable. Regardless of the project's actual position in time, we have information describing the portion of the planned schedule, which should have been accomplished. That is, for a claimed amount of EV at a status point AT, the portion of the PMB which should be accomplished is identified by ES. Another way of describing this relationship is the value of ES indicates where the task performance of the project should be for that amount of duration of the planned schedule. As shown by Figure 2, specific tasks make up that portion of the schedule. The darker shaded areas of the task blocks indicate the portions planned to be completed. If the schedule is adhered to we will observe in the actual performance the identical tasks at the same level of completion as the tasks which make up the plan portion identified by ES. By adhering to the planned sequence of tasks, the manager is assured during project execution that the predecessors to the tasks in work are complete.

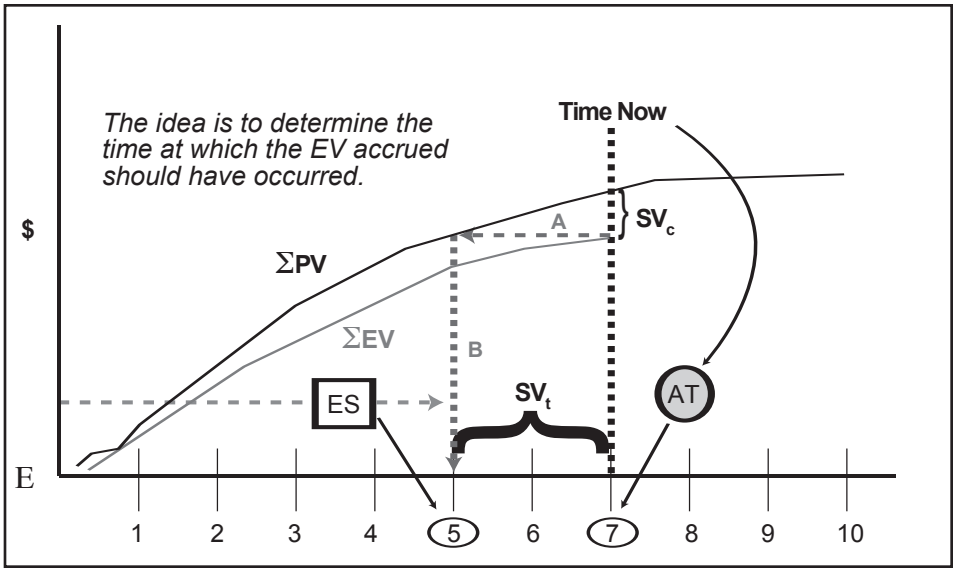


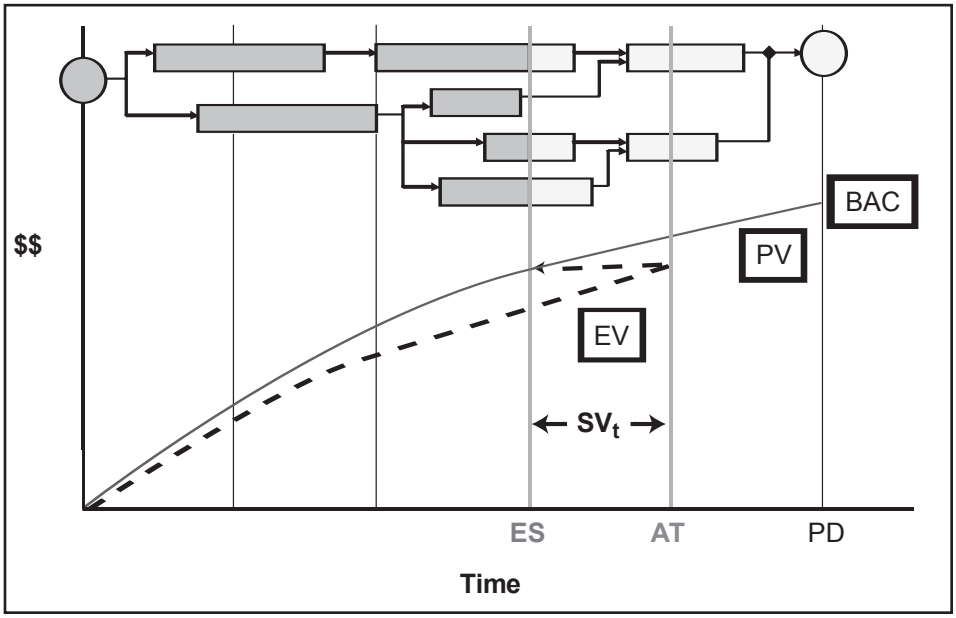
Figure 1: *ES Concept*

It is more than likely the project is not performing synchronously with the schedule; EV is not being accrued in accordance with the plan. As seen in Figure 3 (see page 16), the accumulated EV is the same quantity depicted in Figure 2, but its task distribution is different. Figure 3 is a graphical illustration of the earlier discussion of the reasons for process discipline. The lagging performance for tasks to the left of ES indicates the possibility of a constraint or impediment. Performance may be lagging behind the expectation due to something preventing it from occurring. The EV indicated to the right of ES shows tasks performed at risk; they will likely have significant rework appearing later in the project.

Both sets of tasks, lagging and ahead, cause poor efficiency. Of course, for the lagging tasks, impediments and constraints make progress more difficult. *Concentrating management efforts on alleviating the impediments and constraints will have the greatest positive impact on project performance.*

The darkened tasks to the right of ES indicate performance resulting from

Figure 2: *ES Connects EV to Schedule*



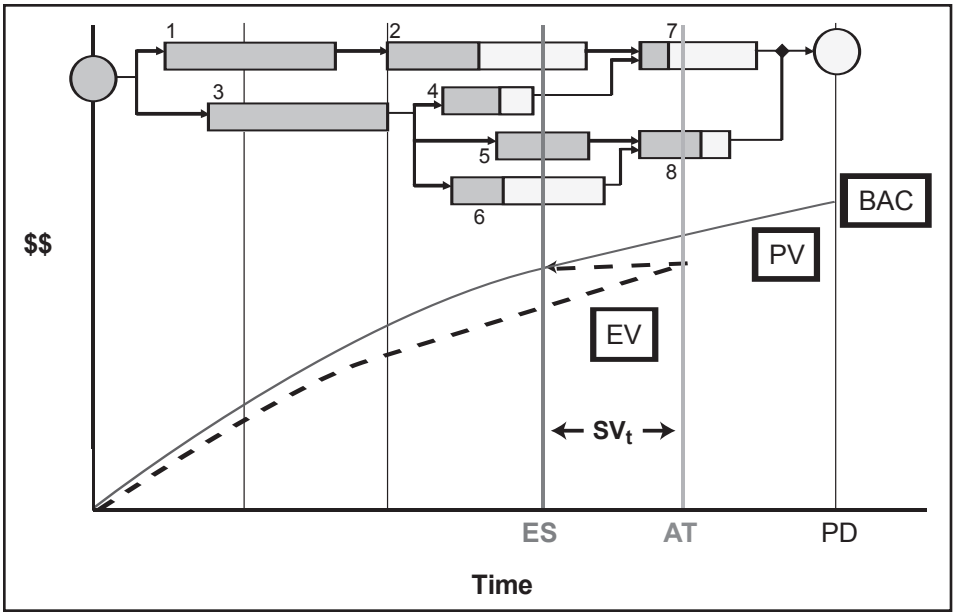


Figure 3: *Actual Distribution of EV*

impediments and constraints or poor process discipline. Frequently, they are executed without complete information. The performers of these tasks must necessarily anticipate the inputs expected from the incomplete preceding tasks; this consumes time and effort and has no associated EV. Because the anticipated inputs are very likely misrepresentations of the future reality, the work accomplished (EV accrued) for these tasks usually contains significant amounts of rework. Complicating the problem, the rework created for a specific task will not be recognized for a period of time. The need for rework will not be apparent until all of the inputs to the task are known or its output is recognized to be incompatible with the requirements of a subsequent task.

This conceptual discussion leads to the measurement of schedule adherence. By determining the EV for the actual tasks performed congruent with the project schedule, a measure can be created. The adherence to schedule characteristic, P, is described mathematically as a ratio:

$$P = \Sigma EV_j / \Sigma PV_j$$

PV_j represents the PV for a task associated with ES. The subscript j denotes the identity of the tasks from the schedule which comprise the planned accomplishment. The sum of all PV_j is equal to the EV accrued at AT. EV_j is the EV for the *j* tasks, limited by the value attributed to the planned tasks, PV_j.

Consequently, the value of P represents the proportion of the EV accrued which exactly matches the planned schedule.

Recall, the question with which we began, *did the accomplishment match exactly the expectation from the schedule*? The P-Factor answers the question and thus is the performance indicator of schedule adherence sought after.

A characteristic of the P-Factor is that its value must be between zero and one; by definition, it cannot exceed one. A second characteristic is that P will exactly equal 1.0 at project completion. P equal to zero indicates that the project accomplishment thus far is not, at all, in accordance with the planned schedule. Conversely, P equal to one indicates perfect conformance.

When the value for P is much less than 1.0, i.e., poor schedule adherence, the project manager has a strong indication the project is experiencing an impediment, the overload of a constraint, or there is poor process discipline. Conversely, when the value of P is very close to 1.0, the PM can feel confident the schedule is being followed and that milestones and interim products are occurring in the proper sequence. *The PM thus has an indicator derived from ES which further enhances the description of project performance portrayed by EVM alone.*

Table 1: *Schedule Adherence Example*

Task	PV	PV at ES	EV at AT	EV – PV	I/C or R
1	10	10	10	0	
2	12	9	5	-4	I/C
3	10	10	10	0	
4	5	5	3	-2	I/C
5	5	2	5	+3	R
6	8	4	3	-1	I/C
7	7	0	1	+1	R
8	5	0	3	+3	R
Total	62	40	40	0	

Example Application

Table 1 contains notional data that relates to Figure 3. The task numbers from the table are identified, as well, in the network diagram of the figure. The total PV for the hypothetical project is 62 units. The total EV accrued at AT is 40 units; the task distribution of EV is beneath the column heading, EV at AT. The task distribution of PV for the ES duration is shown in the PV at ES column.

By calculating the difference, EV minus PV, between the two distribution columns, we can determine which tasks may have impediments or where a constraint has developed. Those tasks are identified by the negative values in the EV-PV column and recorded as a possible impediment or constraint (I/C) in the last column of Table 1; they are tasks 2, 4, and 6. The PM should investigate those three tasks for removal of impediments or alleviation of the constraints.

Should no impeding problem be found, the PM has reason to suspect inappropriate performance by members of the project team, i.e., poor process discipline. It may be discovered that a person assigned one of the tasks identified is insufficiently skilled or trained. *This never happens, does it?* The employee, in order to maintain a satisfactory efficiency for his performance review, executed a downstream task because it was something he knew how to do. *(For this example, the employee is compelled to do the wrong thing. Let us hope that management fully examines the problem and recognizes its own culpability.)*

The column, EV-PV, also indicates positive differences for three tasks: 5, 7, and 8. These tasks are not being performed synchronously with the schedule and are at risk of generating rework, as indicated by the letter R recorded in the table. It is obvious from Figure 3 that tasks 7 and 8 are at risk because some or all of the required inputs to them are absent. However, the risk of task 5 is not so obvious; all of its required inputs are available. With respect to ES, it should be only partially complete. Task 5 completion

is not synchronous with the planned execution at the ES duration. Rework can be generated in this case as well – it is never wise to be too far *out in front*.

To further explain, as the project progresses the detail for task accomplishment becomes much clearer. Oftentimes subtle changes to task requirements are made due to the learning gained during the development process from the prior task accomplishment. By working ahead, the developer unknowingly makes the presumption that his work is unaffected by the other facets of the project. When this occurs, the task worker is not performing synchronously with the plan and the risk of rework is created.

What is the value of the P-Factor for this example? From review of the PV at ES column, the tasks to be included in the calculation are 1 through 6; the sum of PV at ES equals 40. The sum of the EVs in agreement with the PVs is found from the values of tasks 1 through 6 in the EV at AT column. The sum of the values for these tasks is 36. However, recall task 5 is three units ahead of where it should be with respect to the amount of PV planned for that point in time. Subtracting the three units, the EV sum in agreement with the schedule equals 33. As can be seen, another way to calculate the EV in agreement is to add the sum of the negative entries in the EV–PV column to the total EV accrued; i.e., 40 + (- 7) = 33. P can now be calculated as follows:

$$P = \Sigma EV_i / \Sigma PV_i = 33 / 40 = 0.825$$

Thus, approximately 80 percent of the execution is in conformance with the schedule.

Let us presume all of the claimed accomplishment not in schedule conformance requires rework, seven units. For this worst case, nearly 18 percent of the claimed EV must be re-accomplished for the project to complete satisfactorily. Unless this project has considerable reserves, successful completion within the allocated resources is very unlikely. It is obvious; the manager for this project has work to do. However, without the P-Factor indicator and the analysis, it is not so obvious as to what he should investigate and take action to correct.

Real Data

Figure 4 is a graph of the indicators, cost performance index (CPI), SPI(t), and the P-Factor from real project data. For the figure, CPI is the CPI from EVM and the Percent Complete of the x-axis is determined from EV divided by the Budget at Completion (BAC) [1]. As you can see, the schedule adherence (P-Factor) is extremely high, even from the beginning; at 20 percent complete, P is equal to 0.93. The fact that the P-Factor is very nearly 1.0 says that the precedence of the schedule is followed very closely throughout the period of execution shown.

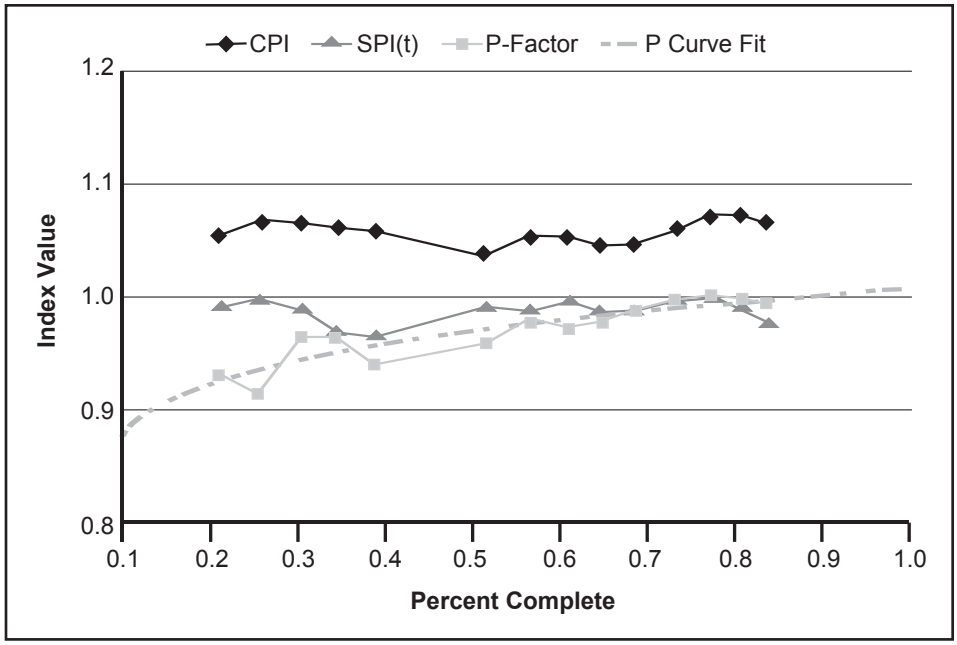


Figure 4: *Project Management Indicators*

Also observed is the curve fit of the P-Factor data points. The curve fit is an illustration of the previous discussion of the behavior of P: as the project percent complete increases, in general the value of P will approach 1.0; at completion, P = 1.0. This behavior is observed with the curve fit line.

The plots of CPI and SPI(t) indicate a very high performing project; CPI hovers around 1.05, while SPI(t) is generally greater than 0.98. The forecast for the project outcome is expected to complete under budget and slightly past its planned completion date. A logical conjecture from the comparison of the indicators is that when the planned schedule is closely followed, output performance is maximized, and the project has the greatest opportunity for success. In other words, when P is a high value, we can expect CPI and SPI(t) to be high, as well. Although this relationship needs verification from further research, the rationale appears reasonable.

Summary

ES is a measure shown over the last four years of application and research examination to provide reliable schedule performance indicators, further enabling duration and completion date forecasting. In this article, the application of ES is extended, thereby facilitating identification of those tasks which should have been accomplished for the EV accrued. From the comparison of the actual distribution of the EV to its planned distribution, it is shown that useful information is available to project managers concerning possible impediments or constraints along with the identification of potential future rework.

The measure for indicating how well the project is following its planned schedule is Schedule Adherence, i.e., the P-Factor. Adhering to the planned sequence of tasks, assures that the predecessors to the tasks in work are complete thereby minimizing the potential for rework. The P-Factor enhances project control capability by providing additional early warning information. When employed with SPI(t) from ES and CPI from traditional EVM, the P-Factor yields more complete project performance information. In turn, the added measure enhances management decision making, and the probability for successful project outcomes.

Final Remarks

Some practitioners of EVM hold to the belief that schedule analysis can be accomplished only through detailed examination of the network schedule. They maintain the understanding and analysis of task precedence and float within the schedule *cannot be accounted for by an indicator*. However, detailed schedule analysis is a burdensome activity and if performed often can have disrupting effects on the project team.

ES offers calculation methods yielding reliable results, which greatly simplify final

duration and completion date forecasting. Furthermore, as described in this article, the development of ES has led to a new and potentially powerful indicator of schedule performance, i.e., Schedule Adherence.

Future research of the proposed Schedule Adherence Indicator is encouraged. To promote experimentation and usage of the measure, the P-Factor calculator is made available for download at <www.earnedschedule.com/Calculator.shtml>.◆

References

1. Lipke, W. "Schedule Is Different." The Measurable News Summer 2003: 31-34.
2. Practice Standard for Earned Value Management. Newtown Square, PA: Project Management Institute, 2005.

Notes

1. The schedule performance indicator from EVM is symbolized by SPI. SPI is equal to the EV divided by the PV at a specific time; i.e., SPI = EV / PV [1].
2. The time-based schedule performance indicator from ES is SPI(t) and is equal to the earned schedule divided by the actual duration (or actual time, AT); i.e., SPI(t) = ES / AT [2].
3. The EVM and ES definitions of SV_e and SV_t, respectively, are as follows: $SV_e = EV - PV$; $SV_t = ES - AT$ [1, 2].

About the Author

Walt Lipke retired in 2005 as deputy chief of the Software Division at Tinker Air Force Base and has more than 35 years experience in the development, maintenance, and management of software for automated testing of avionics. During his tenure, the division achieved several software process improvement milestones, including the coveted Software Engineering Institute/Institute of Electrical and Electronics Engineers award for Software Process Achievement. Lipke has published several articles and presented at conferences internationally on the benefits of SPI and the application of EVM and statistical methods to software projects. He is the creator of the Earned Schedule® technique, which extracts schedule information from EV data. Lipke has a master's degree in physics, and is a member of the physics honor society, Sigma Pi Sigma. In 2007, Lipke received the PMI Metrics Specific Interest Group Scholar Award and the PMI Eric Jenett Award for Project Management Excellence.

1601 Pembroke DR
Norman, OK 73072
Phone: (405) 364-1594
E-mail: waltlipke@cox.net

A Review of Boundary Value Analysis Techniques

Dr. David J. Coe
The University of Alabama in Huntsville

Software testing is an essential element of any software development effort. Developers must have some means of selecting tests to evaluate the completeness and quality of product produced. This article reviews Boundary Value Analysis (BVA), a functional testing methodology that can assist in the identification of an effective set of tests.

Software testing is a fundamental software engineering activity critical to a successful development effort. In fact, an increasingly popular approach to software development is that of *test-driven development* in which tests are identified and documented prior to implementation of the code [1]. The test-driven approach to development places an emphasis on the *quality* of the resulting product by establishing completeness and correctness criteria early. A major challenge to any testing effort is that one must identify a set of tests that are effective at finding defects while keeping the resources associated with applying those tests within project cost and schedule constraints.

The following is an overview of BVA, a systematic methodology for identifying tests to apply. In the following discussion, the term *test case* refers to "a set of inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement." A *test* is defined as either "a set of one or more test cases" or as "the execution of the test cases." A *fault* is "an incorrect step, process, or data definition in a computer program," and a *failure* is the "inability of a system or component to perform its required functions within specified performance requirements [2]." Thus, a primary goal of software testing is to identify failures, which indicate the presence of one or more faults [3].

Overview of BVA

BVA is a black-box approach to identifying test cases. In black-box testing, test cases are selected based upon the desired product functionality as documented in the specifications without consideration of the actual internal structure of the program logic [4]. A fundamental assumption in BVA is that the majority of program errors will occur at critical input (or output) boundaries, places where the mechanics of a calculation or data manipulation must change in order for the program to produce a correct result [3].

An example that illustrates the general concept of boundary values would be a program that calculates income tax for a given income. In a progressive income tax scheme, the tax rate applied increases from low-income brackets to high-income brackets. In this case, the critical input boundaries would be the set of incomes at which the applied tax rate should change along with any minimum or maximum extremes of the income value. Thus, the set of boundary incomes defines the limits of each tax bracket.

Test Case Selection Using BVA

The set of test cases identified by BVA depends upon both the reliability requirements of the software under test and the underlying assumptions on the likelihood of single versus multiple range checking faults. The following discussions of *single-variable* and *multi-variable BVA* are derived from the BVA taxonomy and discussion presented in [3].

Single-Variable BVA

The baseline procedure for BVA begins by identifying the boundary values, typically from the input point of view. All of these boundary values will be incorporated into the set of test cases. In addition to those values, values near the boundaries will be tested. These boundary-adjacent values will help to exercise the program's bounds-checking logic. For example, when testing the range of a value in a branching or looping statement, the developer may use a less-than operator, '<,' when the correct operator should have been a less-than-or-equal to operator, '<=,' or a greater-than operator, '>,' which is adjacent to the less-than operator on most keyboard layouts. Such errors would

Figure 1: *Baseline BVA Test Cases Identified for Single-Variable, Single-Range Example (Shaded area indicates valid values of the variable N)*

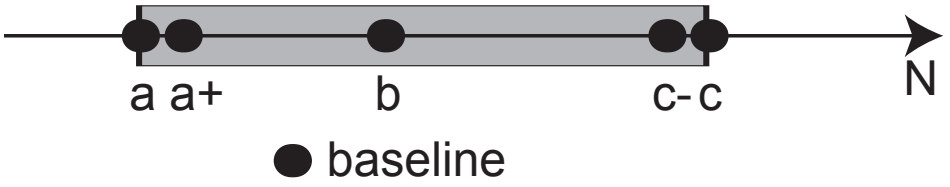


Figure 2: *Single-Variable, Single-Range Baseline Test Cases Augmented With Robustness Tests (shaded area indicates valid values of the variable N)*

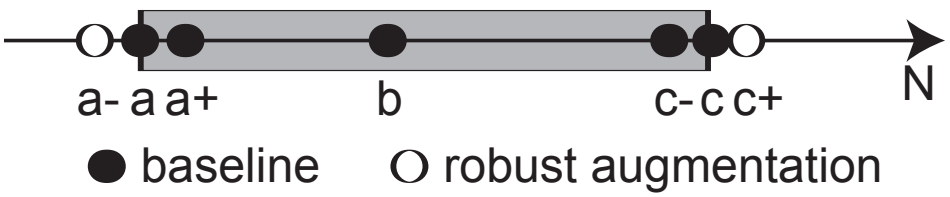
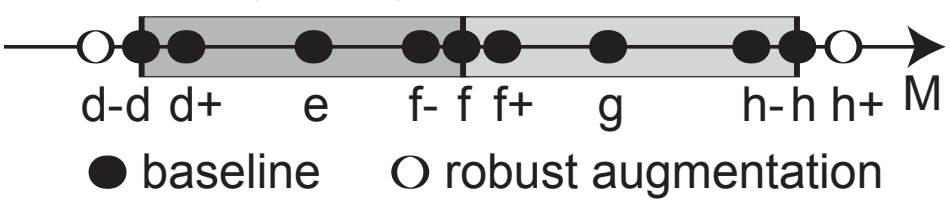


Figure 3: *Single-Variable, Two-Range Test Cases Identified by Robust BVA (Highlighted areas indicate the two subranges of valid values of M)*



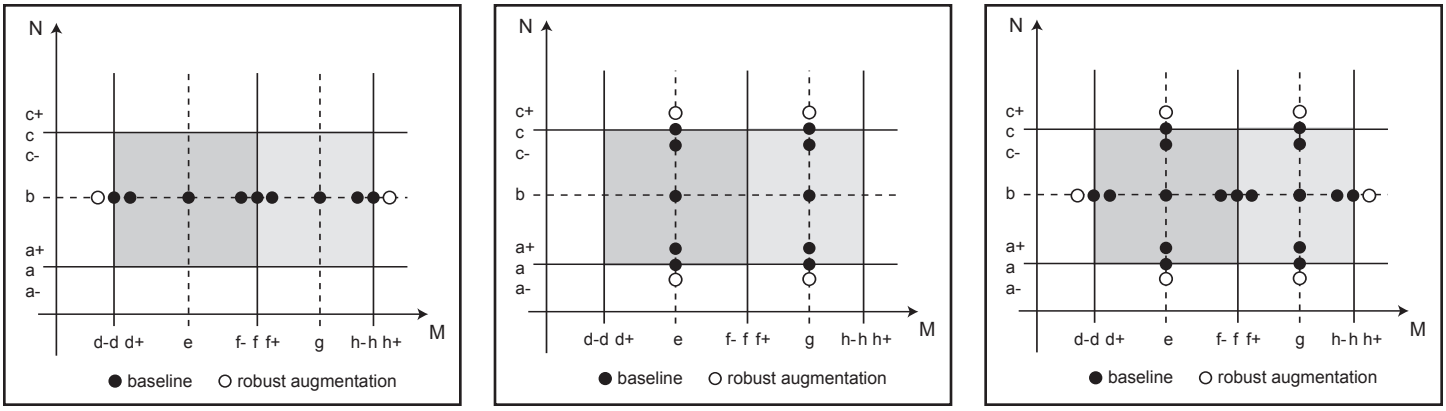


Figure 4 A,B,C: *Single-Fault, Baseline, and Robust Test Cases (A) assuming N is at its nominal value, (B) assuming M is at its nominal values for each subrange, and (C) the set of all test cases identified (derived from* [3]*)*

result in code that compiles but executes incorrectly under certain conditions. To test for these types of errors, values adjacent to the boundary values must be included in the set of test cases. In addition to the boundary and boundary-adjacent values, the baseline BVA procedure includes some nominal value of input (or output) in the set of test cases. The baseline BVA procedure is best illustrated by the following example.

Consider a program with a single input variable *N* that has an output defined only for values of *N* in the range $a \leq N \leq c$. The set of test cases selected would be at *minimum* the set of values $N_{baseline}$ = {*a, a+, b, c-, c*} where *a+* is a value just greater than *a, c-* is a value just less than *c*, and the value *b* is some nominal value that lies between *a+* and *c-*. In this example, the baseline BVA procedure identifies five test cases. As graphed on the number line in Figure 1 (see page 19), the test cases selected under the baseline procedure do not exceed the allowable range of inputs for the variable *N*.

If error handling is critical to the software under test, then one augments the set of test cases identified by the baseline BVA procedure to include robustness tests, that is, values outside the allowable range. The baseline tests identified above are augmented with the values {*a-, c+*} where *a-* is a value just below the minimum acceptable value *a* and *c+* is a value just above the maximum acceptable value *c*. The inclusion of the values {*a-, c+*} in the set of test cases should force execution of any exception handler or defensive code. In this single-input, single-range example, robust BVA identifies a total of seven test cases as shown in Figure 2 (see previous page) where N_{robust} = {*a-, a, a+, b, c-, c, c+*}.

The baseline BVA or robust BVA procedures may also be applied in situations where an input may have multiple subranges. Consider a single input *M* with two adjacent subranges where range #1 is given by $d \leq M < f$ and range #2 is given by $f \leq M \leq h$. The set of test cases would be the union of the test cases identified by applying the BVA procedure to each individual subrange. So, the *union* of test cases resulting from the application of baseline BVA to each subrange individually is given by the following:

$$M_{baseline} = \{d, d+, e, f-, f\} \cup \{f, f+, g, h-, h\}$$
$$= \{d, d+, e, f-, f, f+, g, h-, h\}$$

Application of robust BVA augments $M_{baseline}$ with the extreme values {*d-, h+*} to yield M_{robust} = {*d-, d, d+, e, f-, f, f+, g, h-, h, h+*} as illustrated in Figure 3. The addition of multiple subranges clearly increases the total number of test cases identified. For two adjacent subranges of a single variable, baseline BVA identified nine test cases and robust BVA identified 11 test cases total.

Multi-Variable BVA

The BVA test case selection procedure for multi-variable problems also requires consideration of fault likelihood, what I refer to as a fault model. Under the single-fault model, it is assumed that a failure is the result of a single fault due to the low probability of two or more faults occurring simultaneously [3]. For the multiple-fault model, one assumes that the likelihood of multiple simultaneous faults is no longer insignificant, and thus additional test cases must be selected to address situations such as erroneous range checking on multiple variables simultaneously.

Drawing from our previous single variable examples, assume the single-fault model for a problem that has two inputs, *N* and *M*, with values of *N* in the allowable range $a \leq N \leq c$ and where allowable values of M span range #1, given by $d \leq M < f$, and range #2, given by $f \leq M \leq h$. From our previous discussion, the baseline single variable test cases identified for N and M respectively are the following:

$$N_{baseline} = \{a, a+, b, c-, c\}$$

and

$$M_{baseline} = \{d, d+, e, f-, f, f+, g, h-, h\}$$

Under the single-fault assumption, multi-variable BVA test cases are selected that exercise the boundaries of one variable while the other variables are held at a nominal value. The final set of test cases selected is the union of all test cases identified as this procedure is applied to each individual input in turn. In the following example, I have chosen to apply this procedure to each subrange of each variable in turn to produce a symmetric solution.

Since we have assumed that there are two inputs in this problem, the set of test cases will consist of ordered pairs of inputs (*m,n*) such that *n* is a member of $N_{baseline}$ and *m* is a member of $M_{baseline}$. Figure 4A shows a graph of the nine test cases identified assuming that *n* is held to its nominal value *b* while *m* varies across the members of $M_{baseline}$. The graph in Figure 4B shows the 10 test cases identified in which *m* is held to its nominal value, *e* or g, while *n* varies across the members of $N_{baseline}$. Figure 4C illustrates the union of these sets of test cases. Note that due to the selection of (*e,b*) and (*g,b*) twice, a total of 17 test cases have been identified instead of 19.

For robustness testing, one applies the same procedure starting with the values previously identified in the sets M_{robust} and N_{robust}. Note that under the single-fault model, robustness testing adds only six additional test cases to the 17 baseline test cases for a total of 23 test cases. These additional tests are identified in Figure 4C.

Under the multiple-fault assumption, additional test cases must be selected to detect multiple, simultaneous faults such as erroneous range checking on two vari-

ables at the same time. The multiple-fault BVA procedure again starts with the sets $M_{baseline}$ and $N_{baseline}$ if bounds checking is not critical or M_{robust} and N_{robust} if bounds checking is a high priority. To select BVA test cases assuming that multiple simultaneous faults are likely, one computes the Cartesian product $N_{baseline}$ x $M_{baseline}$ for the baseline multiple-fault test cases or M_{robust} x N_{robust} for the multiple-fault, i.e., worst-case test cases [3].

Given two sets M and N, the *Cartesian product* of M and N is defined as follows:

$$M \times N = \{(m,n) \mid m \in M \wedge n \in N\}$$

where (m,n) denotes an ordered pair [5]. In other words, M x N is the set that consists of all possible ordered pairings of an element from set M with an element of set N. So, if set M contains x elements and set N contains y elements, the resulting set M x N will contain a total x*y total elements.

Figure 5 depicts the baseline and robust BVA test cases identified for our sample problem assuming the multiple-fault model. Note the significant increase in the total number of tests identified. Forty-five baseline test cases were identified for this problem plus an additional 32 for worst-case robustness testing.

Table 1 summarizes the number of test cases identified versus various reliability requirements and fault model assumptions. The multiple-fault assumption significantly increases the total number of tests required, especially in situations where a variable of interest has multiple ranges. Under the single-fault assumption, the incorporation of robustness tests, even in the situation where a variable has multiple ranges, results in a modest increase in the total number of test cases required.

Discussion

From the previous review it is clear that BVA has several advantages: The mechanical nature of the procedure and the symmetry of the tests identified make the BVA procedure easy to remember and use, especially given that critical input boundaries are often already explicitly identified in the requirements. With BVA, one can adjust the number of test cases identified and, thus, the resources expended on testing effort, depending upon the robustness demands of the product.

BVA also serves as an introduction to other test techniques. Discussions of BVA in the literature are often intermingled with a related black-box technique known as Equivalence Partitioning (EP), which utilizes the boundary values in an attempt to define partitions or sets of test cases that are *equivalent* in the sense that all test cases grouped within a particular partition would reveal the presence of the same set of defects and likewise fail to detect other defects. In its simplest form, once the partitions are identified, the set of test cases selected is one representative test case from each partition. A distinct advantage of the EP technique is that the total number of test cases is significantly smaller than the set of test cases identified through BVA. In fact, the set of test cases identified by EP can be a subset of those identified by BVA, and researchers have exploited this fact to reduce the total number of test cases identified in merged BVA-EP schemes.

Studies show, however, that BVA can be effective at identifying failures. In [6], Reid investigated the effectiveness of random testing, equivalence partitioning, and boundary value analysis techniques.

> *"With BVA, one can adjust the number of test cases identified and, thus, the resources expended on testing effort ..."*

According to his results, the probability that BVA would detect a fault was more than six times higher than random testing and more than twice as high as equivalence partitioning. The cost for this increased effectiveness was additional test cases, on the order of two to three times the number of test cases as equivalence partitioning depending upon the particular variations of the techniques employed.

Other studies have compared functional, structural, and code reading test methodologies. In structural testing, test cases are selected to exercise specific program elements such as statements, branches, or paths through a code segment. For example, to achieve 100 percent statement coverage, the set of test cases identified must force execution of each program statement at least once. For 100 percent branch coverage, the set of test cases identified must force each branch option to execute at least once. For code reading, individuals were given the source code and asked to work backwards towards a specification for that program by successively grouping subprograms into logical modules until an understanding of the overall functionality was achieved. Failures were detected by comparing the actual specification to that derived by the code reader.

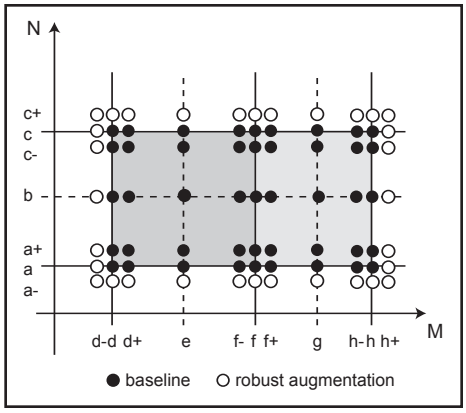


Figure 5: *Multiple-Fault, Baseline, and Robust Tests (derived from* [3]*)*

Basili and Selby [7] studied the relative effectiveness of a combined BVA and EP functional testing approach against 100% statement coverage structural testing and code reading. Among professional programmers, they found that code reading detected the most faults followed by functional testing and then structural testing. The average maximum statement coverage achieved by both the functional and structural testers was 97 percent yet the functional testing approach detected more faults than did structural testing in this study. It was also noted that the number of faults detected varied with the type of software tested, and that the testing techniques tended to detect different types of faults.

The relative effectiveness of a combined BVA and EP functional testing technique, 100% branch coverage structural testing, and code reading were compared in [8]. This study also observed that the effectiveness of the techniques varied with both the nature of the programs and of the faults themselves. Most importantly, this study determined that the use of two or more test techniques together, such as functional testing and code reading, was more effective in general than any single methodology since the techniques were

Table 1: *Number of Test Cases Identified for Two-Variable BVA Problem*

Number of Tests Identified		Assumed Fault Model	
		Single-Fault	Multiple-Fault
Reliability Requirement	Baseline	17	45
	Robust	23	77

essentially complementary [8].

Conclusions

The BVA technique provides a systematic procedure for evaluating the completeness and quality of a software product. While some may find excessive redundancy in the set of test cases generated by boundary value analysis, I have found that the symmetry and mechanical nature of BVA help to make the procedure both easier to teach at the undergraduate level and easy to remember and apply in practice. BVA also provides a basis for learning other techniques, in particular, equivalence partitioning, and it is effective as a functional testing technique for identifying failures. Empirical studies show, however, that a combination of functional, structural, and/or code reading techniques is generally more effective than relying upon any single methodology since the effectiveness of the techniques vary with both the type of code being tested and the nature of the faults.◆

References

1. Schach, Stephen R. Object-Oriented and Classical Software Engineering. 7th ed., McGraw Hill, 2007.
2. IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard 610.12-1990.
3. Jorgensen, Paul C. Software Testing: A Craftman's Approach. 2nd ed. CRC Press, 2002.
4. Perry, William E. Effective Methods for Software Testing. 3rd ed. Wiley Publishing, 2006.
5. Beyer, William H. CRC Standard Mathematical Tables. 25th ed. CRC Press, 1981.
6. Reid, S.C. 1997. An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing. Proc. of the 4th International Symposium on Software Metrics 05-07 Nov. 1997, Washington, D.C.: IEEE Computer Society, 1997.
7. Basili, Victor R., and Richard W Selby. "Comparing the Effectiveness of Software Testing Strategies." IEEE Trans. on Software Engineering 13.12 (1987): 1278-1296.
8. Wood, M., M. Roper, A. Brooks, and J. Miller. Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study. ACM SIGSOFT Software Engineering Notes, Proc. of the 6th European conference held jointly with the 5th ACM SIGSOFT International Symposium on Foundations of Software Engineering ESEC '97/FSE-5, 22.6 (1997): 262-277.

About the Author

David J. Coe, Ph.D., is an assistant professor in the department of electrical and computer engineering at the University of Alabama in Huntsville where he teaches undergraduate and graduate courses in C++ programming, data structures, and software engineering. He has consulted locally in the areas of software engineering and software process. Coe has an undergraduate degree in computer science from Duke University, and a master of science degree in electrical engineering and doctorate degree in electrical engineering from the Georgia Institute of Technology.

The University of Alabama in Huntsville
Department of Electrical and Computer Engineering
217-F Engineering BLDG
Huntsville, AL 35899
Phone: (256) 824-3583
E-mail: coe@ece.uah.edu

VoIP Softphones

David Premeaux
U.S. Army Information Systems Engineering Command

Voice over Internet Protocol (VoIP) provides the user with an opportunity to combine the use of a telephone with a personal computer (PC) into what is known as a Softphone. A Softphone allows users to place and receive calls using a PC. This article covers what a Softphone is and its issues, such as quality of service and security, which affect Softphones. The Technical Integration Center (TIC) currently does not recommend significant use of Softphones in the Army due to security and certification issues.

In the past 10 years technology has advanced to the point whereby telephone calls can be placed over Internet Protocol (IP) packet networks, also know as VoIP. One of the developments in this transition to VoIP was to turn a computer into a VoIP telephone by loading and running a VoIP software application on the computer. This VoIP application has emerged to be called a Softphone. A key motivation for using the Softphone is lower cost. This is due to the fact that the Softphone is little more than software, as compared to a traditional telephone that is mostly or all hardware. Softphones are also able to take advantage of making calls over the Internet with little additional equipment. This can save on long distance charges, especially when talking to another Softphone. Other advantages of the Softphone include potential integration with other applications, no space needed on the desk for a telephone, and the ability to move one's phone number with a computer.

What Is a Softphone?

For this article, a Softphone is a VoIP client application running on a computer. The Softphone uses VoIP signaling to establish calls, tear down calls, and take advantage of call features such as call forwarding. The Softphone also uses VoIP protocols to transport audio traffic in IP packets to another VoIP device. The Softphone is a client device, as it is the user device for establishing and tearing down calls. Other applications, such as a call processor application running on a computer, would not be considered a Softphone. The Softphone is also a software application loaded onto a computer and not a hardware device running in a computer.

Softphones using the Microsoft operating system will generally use the Telephony Application Programming Interface (TAPI), which enables PCs to support telephone services. TAPI provides support for such features as the volume control, microphone level, speakerphone, call control, etc. The version in Extra Professional also provides support for telephones connected to a PC via a Universal Serial Bus port.

The most common motivation for using a Softphone is avoiding long distance telephone calls. People using them for business can connect up from a hotel room and place calls back to the office using a PC and avoid using the hotel telephone or cell phone minutes. Home users are able to call and talk to each other using PCs (sometimes with video added) and avoid toll charges.

For the Department of Defense (DoD), Softphones have potential applications with tactical users. A user could gain telephone service simply by connecting a PC to an IP network and be able to place calls without a local call processor set up. An added advantage is that the user's telephone number would move with the PC, making the user more reachable.

Operational Aspects of Softphones

Operation of a Softphone is significantly different from the operation of a traditional telephone. In order to place/receive calls at any time, the user's computer must be turned on and the Softphone application running all the time. Power must also be provided to the computer at all times, and in the event power is lost, the computer needs to be re-booted. This can be avoided by providing power backup to the computer in the form of an uninterruptible power supply. The Softphone will also only be as reliable as the computer. If the computer is not stable and has to be rebooted periodically, the reliability of the Softphone will be affected.

Most traditional telephones have a handset the user utilizes for talking and listening. Most Softphone applications either use the computer speakers and a microphone or use a headset that includes both an earpiece and microphone. Answering a call with a traditional telephone is done by picking up the handset; whereas a Softphone is answered by clicking on an *answer call* icon. Likewise, ending a call with a traditional telephone is typically done by putting the handset into the cradle; whereas a Softphone call is ended by clicking on an icon to end the call.

Call features also work differently, and this is one of the areas where Softphones have an advantage over traditional telephone sets. With a traditional telephone call features are activated by selecting different combinations of digits. For example, to have calls forwarded a user might have to dial the digits #75 and then the call transfer number. With a Softphone, the user would select the call transfer icon and then enter the call transfer number. This eliminates the need to remember or look up various digit combinations to enable call features. A number of vendor implementations of Softphones allow the graphical user interface (GUI) on the Softphone to be used with a traditional telephone. Each user has a computer with the GUI loaded and a separate telephone. The telephone is used as a traditional telephone, but when the user wants to utilize a call feature, such as forwarding a call, it is done on the GUI interface.

Softphones also have the advantage of integrating well with other applications. For example, the Microsoft Netmeeting application can place calls, but it can also share out an application between users. This would enable two users to hold a conversation and share a Word document they would both be able to see and change. Other applications that can be integrated with the Softphone are Video Teleconferencing and whiteboards, which allow both sides to write on a virtual *chalkboard* and each can see what the other is drawing. A new feature forthcoming to the Web is a Softphone built into a Web site. A user could read a Web page, have a question, and click on a link that would provide audio communication with someone at customer service. This ability to integrate with applications also makes Softphones ideal for call centers. A worker in a call center could have a conversation with a customer while other applications integrated with the Softphone could bring up information on the customer.

Softphones have the ability to call other Softphones on the Internet or place calls to the Public Switched Telephone Network (PSTN). Softphones can contact each other directly over the Internet a couple of ways. One way is to have the calling party *dial* the IP address of the called party and establish a connection. Another way is to register with a service. The service provides either a telephone number or name that is put into a registration server along with the user's IP address when the user registers. The calling

party receives the called party's IP address using the registration service and establishes the call. It should be pointed out that Network Address Translation (NAT) can cause problems for Softphones connecting directly, and this will be covered later in more detail.

Softphones can also be set up to make calls to the PSTN. This is done as part of a VoIP solution that includes a VoIP gateway with connectivity to the PSTN. A popular option in the commercial world is to pay for a service that includes a gateway to the PSTN. When the Softphone connects to the PSTN, it will need to have either a real telephone number or an extension number. The service provides the means of registering the telephone number with the user's IP address.

When a Softphones is loaded onto a laptop computer it has the added advantage of being mobile. It still has the ability to connect peer-to-peer or to its PSTN service provider when its location has changed. One interesting feature of a mobile Softphone is that its telephone number moves with it. For example, if a user is connected with a Softphone to the Internet in Dallas and has a Dallas telephone number, and that user disconnects, goes to Denver and connects to the Internet there, then the user's telephone number will appear to be the number from Dallas. If someone calls the user's Dallas telephone number, the Softphone in Denver will ring. This adds an element of convenience to the Softphone, but also has an effect on 911 service.

911 service is designed to map the user's telephone number to a location. When a user dials 911, the operator is able to query a database and determine location from the user's phone number. When a Softphone has a telephone number assigned and stays in one location, there is no issue with mapping this number in the database to the location. When the Softphone has a phone number and changes location, this can pose a problem. If the user in the previous example were to dial 911 while in Denver, the call would be answered by an operator in Dallas, who would assume that the user was in Dallas. This could have a serious impact on emergency services. A law was passed recently that requires commercial providers of VoIP service (including Softphones) to offer users with a means of providing their location information. This only applies to the PSTN connection services and not to the peer-to-peer services. The U.S. Army system has not provided such a number-to-location Softphone service, and it is recommended that 911 calls be placed using Softphones only as a last resort.

Technical Aspects of a Softphone

This section will discuss how a Softphone works and the protocols it uses for communication. Figure 1 shows a typical VoIP configuration that includes a Softphone and a PSTN gateway. For the peer-to-peer case, the configuration consists only of two or more Softphones connected to an IP network.

The Softphone uses the registration server to register its user name (typically a telephone number or Universal Resource Identifier to IP address mapping). Registering will require some form of authentication, such as a personal identification number or Common Access Card. The IP connection between the Softphone and the registration server should be encrypted to protect authentication information.

The call processor is used for establishing calls, tearing down calls, routing calls, and supporting call features. The Softphone sends and receives call signaling messages from the call processor. The Softphone uses the call signaling messages to establish calls to the other devices, including gateways, IP telephones, and other Softphones.

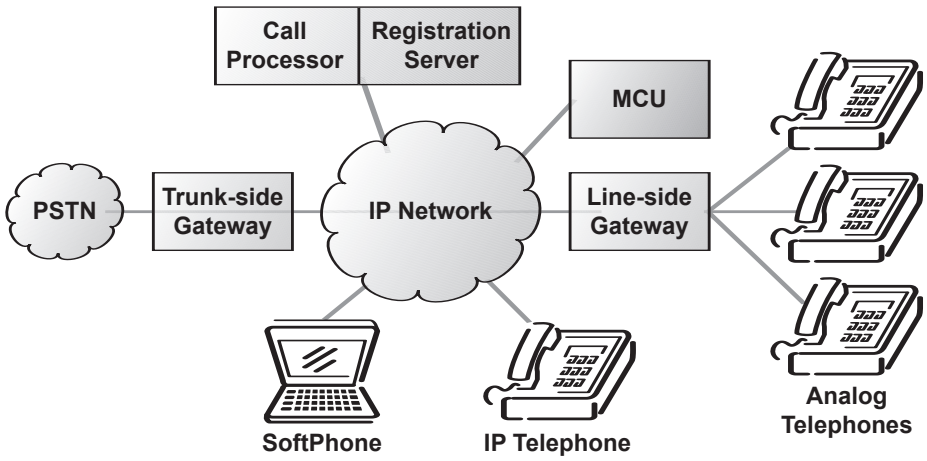
Call signaling messages currently used today include H.323 and Session Initiation Protocol (SIP). The H.323 and SIP protocols are designed to have the intelligence of the VoIP system pushed to the edge, enabling them to call each other without requiring a call processor. The H.323 protocol was developed by the International Telecommunications Union and is the oldest of the protocols and currently the most heavily implemented [1]. The SIP protocol was developed by the Internet Engineering Task Force (IETF) and is considered *lighter* from a code implementation and processor perspective [2]. The SIP protocol is becoming more popular and is expected by many to replace H.323 in the future.

The actual audio traffic flows between the Softphone and other devices in packets that use the Real-time Transfer Protocol (RTP). These packets contain the audio, as well as timing information, sequence numbering, an identifier of the information (compressed voice, video, etc.), and other information. There is also a secure version of RTP available, called Secure RTP, which provides encryption and authentication of the voice traffic. The Real Time Control Protocol supports RTP by conveying information about the quality of the communication, such as jitter and packet loss.

Quality of Service (QoS) is one of the major issues for Softphones. When data traffic experiences packet loss or significant delay, the packets are present and the user observes that the data is taking longer to send or receive. For VoIP traffic, significant packet loss, delay, or jitter (variations in delay) is noticeable to the user. Resending lost packets is not an option, as the conversation will have moved on by the time they are retransmitted. QoS solves these problems by enabling voice packets to get queuing priority over data packets in the IP network.

The Softphone sets QoS and tells the IP network it needs priority in a couple of ways. The first way is by setting the Diffserv bits in the IP header to a higher priority. Layer 3 Ethernet switches will look at these bits and put these packets into a higher priority queue. Another way is to set the Institute for Electrical and Electronics Engineers (IEEE) 802.1P priority bits, which are sent in the IEEE 802.1Q virtual local area network (VLAN) tag. Layer 2 Ethernet switches look at these bits and use them to prioritize the packets. The VLAN tag also has a significant role in logically separating the voice and data traffic, with voice traffic receiving one tag value and data traffic another. Both of these methods of providing QoS work fine in DoD local area networks (LANs), but they are currently not supported in the Non-secure Internet Protocol Router Network or in the commercial Internet. This means Softphones used in a remote fashion will not have any

Figure 1: *Softphone in a Typical VoIP Configuration*



QoS and its traffic will receive no priority.

A problem with QoS and Softphone is the need for QoS within the computer. Computers are generally not set up to provide priority on the internal buses and interfaces to certain applications. There is a means to provide priority to an application within Windows, but this tends to make the system unstable. As a result of the lack of QoS, the latency in the Softphone can be on the order of hundreds of milliseconds, which are at a level where the human ear can begin to detect it and is outside the 60 milliseconds DoD end-to-end VoIP limit.

Another technical issue for Softphones is circumventing the NAT point. When all of the VoIP devices are connected in the same LAN this is not an issue. However, when the calling party is on one side of a NAT point and the called party is on the other, there is a problem. The signaling message the calling party sends to the called party contains the IP address of the calling party. When packets pass through the NAT point this IP address is changed. When the called party attempts to send packets to the IP address in the signaling message, they are dropped (especially if private addressing was used). One current solution is to use the Simple Transversal of User Datagram Protocol through NAT protocol. This protocol works by having the Softphone communicate with a server outside the NAT point. The server is able to see its real IP address and port number and communicate this back to the Softphone. The Softphone then uses this IP address and port number in its signaling messages.

The VoIP devices, including Softphones, have a few tricks for reducing the amount of bandwidth that they utilize. One of them is to use voice compression algorithms. Uncompressed voice (G.711) uses 64 Kilobits per second (Kbps) plus IP network overhead. Other algorithms, such as G.729 (which uses eight Kbps), use less bandwidth. The drawback is that voice quality may be affected. Current DoD policy only allows G.711, but this is expected to change in the future, especially when VoIP goes to tactical units. Another trick is to use Voice Activity Detection (VAD). When a Softphone uses VAD it only sends voice packets when the user is talking. No packets are sent that contain silence. In a typical conversation, only one person is talking at a time so there is audio in one direction and silence in the other. When the silence packets are removed, the amount of bandwidth utilized can be reduced by 50 percent or more. One feature to look for in a Softphone that uses VAD is background noise insertion. Without this, the telephone connection sound is so quiet during periods of silence removal it appears the connection is dead.

An issue for VoIP and Softphones in the future will be Internet Protocol Version 6 (IPV6). Currently, all DoD IP networks are expected to be capable of transitioning to IPV6 by 2008. The computer, the Softphone application, and the operating system will need to support IPV6 for the Softphones to use IPV6. For the Softphone to work with the other VoIP devices, the call processor/registration server, gateways, IP telephones, etc. within its enclave will all need to be running IPV6. The IPV6 protocol may also have an impact on the NAT problem. Due to the large address space of IPV6, it is anticipated that IPV6 will make NAT unnecessary.

Security Issues With Softphones

Security is currently the most difficult issue to overcome with Softphones. The current Defense Information Systems Agency *Security Technical Implementation Guide* (STIG) states, "The use of Softphones is highly discouraged." This is due to a number of items related to the nature of Softphones [3]. This section will go into these, along with the STIG requirements, in more detail.

For VoIP implementations, the security requirements require that the voice and data traffic be separated into networks, either physically or logically. Separate physical networks require separate networking devices, such as switches and routers, for both data and voice networks. Logical separation means that the traffic is separated into logical networks, typically using VLANs. Data devices are connected to data network devices or ports in the data network VLAN and likewise for the voice devices. The major issue with Softphones is they tend to reside on computers having applications requiring access to both data and voice networks. For example, the Softphone computer would have the Softphone application, and then it might have other applications, such as e-mail, Web browsing, etc., that require access to the data network. The following requirement in the STIG addresses this issue:

> (VoIP0150: CAT I) The Information Assurance Officer (IAO) requirement will ensure that if/when approved Softphones are used in the LAN, the following conditions are met:
> - The host computer contains a Network Interface Card (NIC), (commonly called a network adapter) that is 802.1Q (VLAN tagging) and 802.1P (priority tagging) capable.
> - The host computer, NIC, and IP Softphone agent software is configured to use separate 802.1Q VLAN tags for voice and data.
> - Alternatively, dual NICs may be used where voice traffic is routed to one NIC and data traffic is routed to the other. Each NIC is connected to an access switch port residing in the appropriate VLAN.
> - The host computer will be connected to separate voice and data VLANs that have been created expressly for the Softphone host(s). That is to say that the LAN should have a voice VLAN and a data VLAN dedicated to hosts with IP Softphone agents installed. [3]

A couple of issues occur with implementing these requirements. The first is that most computer NIC cards are not able to support VLAN tagging. This would make two NICs in the computer necessary. The second is that some means need to be in place to ensure that the voice traffic only goes to the voice VLAN and the data traffic only goes to the data VLAN. The major security concern here is a hacker coming into a computer on the data network and routing over to the voice network.

The STIG also addresses the case where the Softphone is used in a computer that is accessing the network remotely. The STIG states the following:

> (VoIP0160: CAT I) The IAO will ensure that if/when approved Softphones are used in remote connectivity situations, the following conditions are met:
> - The host computer connects to the "home LAN" through a Virtual Private Network (VPN) connection.
> - The VPN is terminated at the enclave boundary in accordance with the Enclave STIG.
> - The voice and data traffic is routed appropriately to separate voice and data VLANs in the "home LAN."
> - The IP Softphone agent connects to the Call Manager (call processor) on the "home LAN" through the VPN using "home LAN" IP addressing. [3]

Implementing this has the same issues as connecting locally, namely keeping the voice and data traffic separate. This is harder to do remotely, as the remote computer would need to tag the traffic appropriately and put it into a VPN. There would also be QoS and Joint Interoperability Test Center (JITC) cer-

tification issues with using Softphones remotely (this is discussed in the next section).

The STIG also provides the following guidance when Softphones are used in a call center:

> (VoIP0165: CAT I) The IAO will ensure that, if/when approved Softphones are used in a call center situation; the call center network is configured as a separate enclave and secured in accordance with all applicable STIGs.

This means that the call center VoIP traffic must be separated, either physically or logically, from the rest of the IP traffic, in addition to complying with all of the other STIGs.

Due to the security issues with Softphones, the STIG also provides the following guidance to Designated Approving Authorities (DAAs):

> (VoIP0130: CAT I) The IAO will ensure that written DAA approval is obtained prior to the use of any IP Softphone agent software. The IAO will maintain documentation pertaining to such approval for inspection by auditors.
>
> (VoIP0135: CAT I) The IAO will ensure a local IP Softphone policy exists and is being enforced that addresses the following:
> - Prohibits the installation and use of IP Softphone agent software on workstations (fixed or portable) intended for day-to-day use in the user's normal workspace.
> - Prohibits the use of IP Softphone agent software in the user's normal workspace, which has been approved and installed on a portable workstation for the purpose of VoIP communications while traveling.
> - Prohibits the installation and use of IP Softphone agent software clients that are independently configured by end users for personal use or that is provided by commercial Internet Telephony Provider service providers.
> - Requires prior justification and DAA approval for the use of any IP Softphone agent software.
> - Requires that the justification and DAA approval of IP Softphone agent software use is reviewed annually and approval renewed if justified.

JITC Certification Issues

Public law and DoD policy requires that all voice solutions attached to the Defense Switched Network or PSTN obtain interoperability and become Information Assurance certified. For VoIP, the voice solution includes the call processors, registration servers, IP telephones, gateways, and Softphones. While a number of VoIP solutions currently are certified, none of them include a Softphone. This is partly due to difficulty in meeting the security and QoS requirements and partly due to the question of configuration change. The DISA/JITC policy requires a VoIP solution to be recertified if its configuration changes from what was certified. How this would affect Softphones is not yet known. For example, if a computer with a certified Softphone were to change its audio card to a different brand, would it need to be recertified? There is currently no experience with this issue.

There is currently a disconnect in DoD policy regarding the use of Softphones from a remote location, such as a hotel room. The STIG allows it under certain circumstances; whereas, the DISA General Switching Center Requirement (GSCR) (which contains the requirements for interoperability certification) requires end-to-end QoS and a certification of the entire network the VoIP traffic will be traversing [4].

One of the features that a Softphone would need to support to obtain JITC certification for command and control (C2) users is MultiLevel Precedence and Preemption (MLPP). The MLPP allows a caller with a higher precedence to preempt a call of lower precedence. This is typically used when high priority calls need to get through and lines are tied up.

Currently, all JITC certified solutions consist of a LAN for the IP network. The use of VoIP across the wide area network and between services has not been worked out. Currently, if there were a certified configuration that included a Softphone, the Softphone would need to go to a PSTN gateway in order to place a call off of an installation.

Conclusion

While IP Softphones offer several advantages, including mobility and a GUI for call features, it may be a number of years before they are common in DoD telephone systems, with the possible exception of call centers. This is due to a number of reasons. Softphones are still awkward to use due to the lack of a handset. Security and QoS issues will make them difficult to implement and secure. The lack of location awareness when used as a mobile device makes them risky for 911 use. Until JITC certifies a VoIP solution that includes a Softphone, it will be a violation of DoD policy to use one.

Recommendations

The U.S. Army Information Systems Engineering Command (USAISEC) Technology Integration Center (TIC) recommends a continuing effort to examine Softphones, especially in applications such as call centers. Due to the technical complexities of complying with security and performance requirements, we do not recommend any significant move to replace traditional telephones or IP telephones with Softphones at this time.◆

References

1. International Telecommunications Union. "H.323 Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service." Nov. 1996.
2. IETF. Request for Comment 3261, Session Initiation Protocol, June 2002.
3. DoD. Voice over IP STIG, V2R1, 29 Aug. 2005.
4. U.S. Department of Defense, Voice Networks Generic Switching Center Requirements (GSCR), Sept. 2004.

Disclaimers

1. Approved for public release; distribution is unlimited.
2. Disclaimer: The use of trade names in this document does not constitute an official endorsement or approval of the use of such commercial hardware or software. Do not cite this document for advertisement.

About the Author

David Premeaux is the USAISEC Critical Skills Expert for Networking Technology for the TIC at Fort Huachuca, Arizona.

U.S. Army Information Systems Engineering Command
Technology Integration Center
ATTN: AMSEL-IE-TI
Fort Huachuca, AZ 85613
Phone: (520) 533-2867
DSN: 821-2867
Fax: (520) 533-5676
E-mail: david.premeaux@us.army.mil

Truth and Confidence: Some of the Realities of Software Project Estimation

Phillip G. Armour
Senior Consultant Corvus International, Inc.

Software project estimation is not what we think it is because, to some extent, software is not what we think it is. This article explores an alternative view of both software and project estimation and concludes that the process of estimation could be much more valuable than we usually make it.

Predicting the future can be a rewarding occupation, but it can also be a dangerous one. Historically, oracles were often praised and lauded. But they were also stoned to death if they were wrong – and sometimes if they were right [1].

Software project estimation is a difficult task for a simple reason: Software is not really a product, it is a packaging of knowledge and we cannot measure knowledge. Software is best thought of as a knowledge storage medium rather than a manufactured product [2]. It is one of the five places we can put knowledge once we have obtained it, the other four being (in historical order): DNA (Deoxyribonucleic Acid), brains, hardware, and books. However, knowledge in software has different characteristics than knowledge stored in the other media [3].

Shooting Tanks

During World War II, the knowledge of how to hit a tank with a bazooka was stored in several places: It was stored in military manuals (the book form) and in the ranging and sighting device (the hardware form), but it was stored mostly in the operator's head (the brain form) (see Figure 1). There are some drawbacks with these media: The manual only *describes* the knowledge – it does not actually do anything; the sighting mechanism allows for storage and use of only a few of the variables – mostly the distance-to-target versus elevation relationship; and the brain-resident knowledge has the distinct disadvantage that the soldier could be shot at while attempting to hit the target. In modern weapons systems, we have moved almost all of this knowledge into the missile in an active software-resident form.

Not Product Producing

If software is not a product, but is a medium, then software development is not a product-producing activity. In fact, it is best thought of as a *knowledge acquisition* activity. Most of the effort on a software project is related to acquiring and validating knowledge rather than creating a product. We know what we are doing.

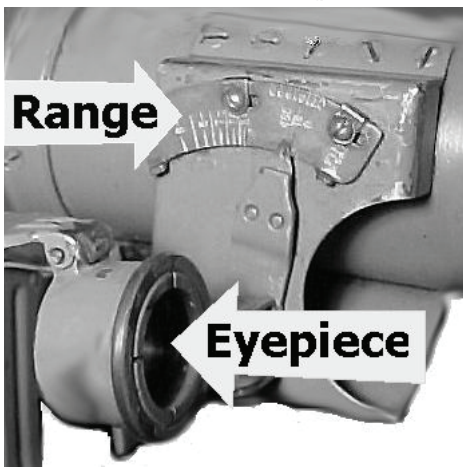
In an attempt to estimate projects, we are trying to figure out how much knowledge we do not have and how much time and effort it will take to get it, plus a small amount of time and effort to translate it into the executable form once we have obtained it. There are two challenges to this: First, we are trying to measure something we *do not* have which is always hard to do, and second – and very importantly – we are trying to measure knowledge, and knowledge is simply not a measurable thing.

This leads us to some observations about the essential nature of project estimation:

- **We cannot have an accurate estimate.** Apart from it being an oxymoron, there is a simple reason why estimates cannot be accurate – we simply do not have the data or knowledge we need to be accurate. The primary activity of a software project is to get this knowledge. The only point in time where we can reasonably assert we are accurate is at the end of the project when we have acquired all the knowledge and resolved all the uncertainty.

  It is possible to have a *lucky* estimate. This happens when all the things we did not or could not think of that slowed the project down and all the other things we did not think of that speeded the project up happen to be equal. Since there are more things that will slow a project down than speed it up – an application of the 2nd Law of Thermodynamics to projects – we usually underestimate.
- **The purpose of estimating is not to come up with an end-date.** This is usually what we are asked for when someone wants an estimate, but it is not a well-formed request. For most projects there is a wide range of possible dates when the project might finish (see Figure 2, page 28). At the point in time when we produce the estimate, we can posit a trade-off of *probability of success* for schedule and other resources. It is easy to be 100 percent successful in projects – simply take a very long time and use a very large number of very good resources. In reality, the purpose of estimation is not to deduce an end-date, it is to derive the *probability function* that describes the range of viable end-dates. The project completion date and schedule is not determined by estimation but by the commitment process.
- **Estimation is not commitment.** Making an estimate is not the same thing as making a commitment. The job of estimation is to identify the project's probability function. The job of the commitment process is to select the point along the probability function that best manages the risk/return ratio. Estimation is a technical activity; commitment is a *business* activity, and they operate on quite different data.
- **The project estimate may not be dependent on the delivered system size.** Despite the fact the every estimation process used in software development operates on the expected delivered system size, the relationship can be quite tenuous. The final system size may be an *indicator* of the effort necessary to develop the system. All else being equal, if one system expects to have twice as many (say) lines of code as another, that

Figure 1: *Bazooka Sighting Mechanism. Photo property of <www.antiquefirearm.com> and <www.andrax.com>. Used with permission*



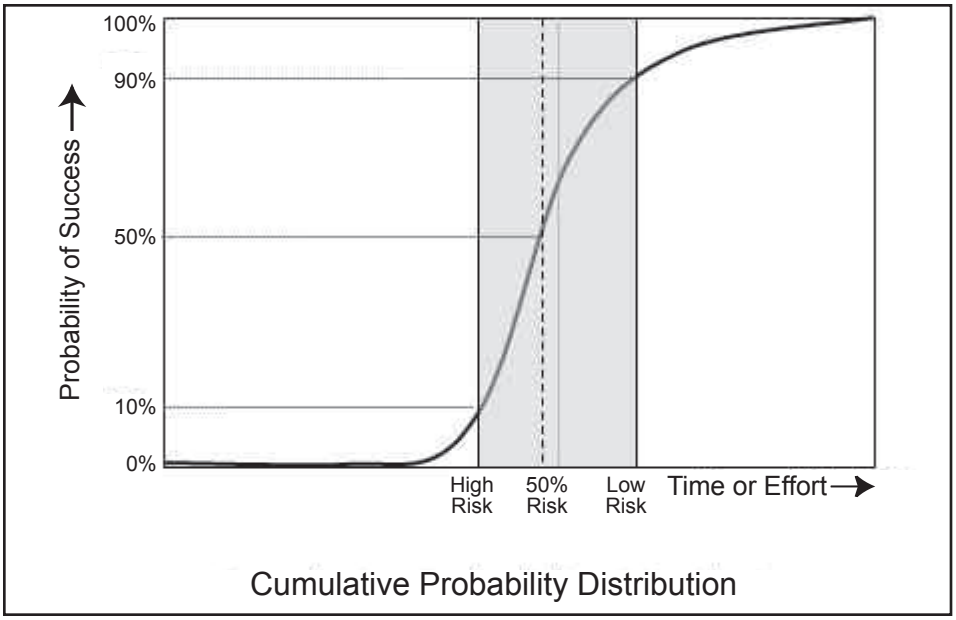


Figure 2: *Probability of Completion*

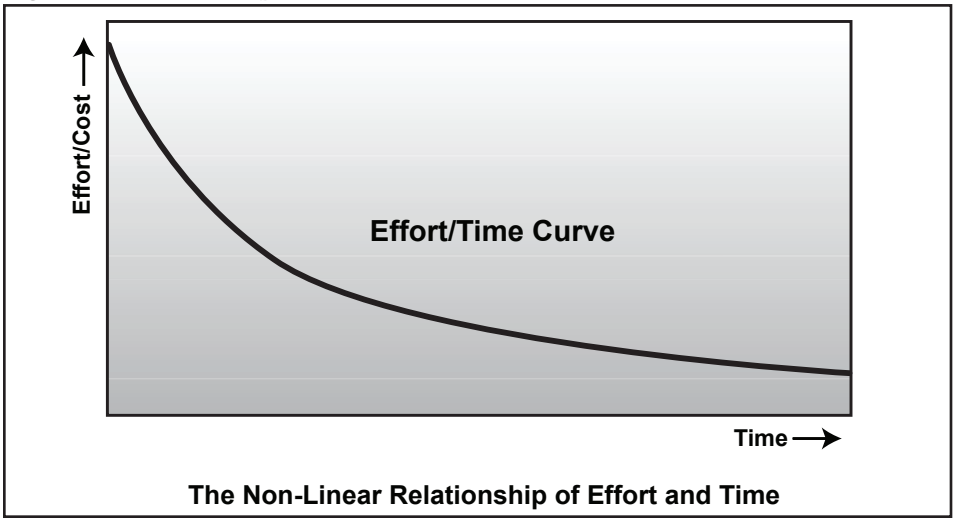
system will require proportionally more time and effort. The trouble is, all else is rarely equal. Viewed as a knowledge acquisition activity, it is clear why project effort may not have much to do with the final size. If we use experienced developers, they do not have to acquire as much knowledge to produce a system of a given size as less experienced developers, but the system size does not change. If we can reuse code, either from a library, or embedded within a language, the effort is less, since some of the knowledge is already stored in an accessible software medium.

Also, a system may be small in terms of its line-of-code form, but it may have very high *knowledge density* as is true of real-time embedded systems – the amount of knowledge needed to make them work divided by the final executable size is much higher than for typical business systems. However, while the knowledge density of an information technology system might be light, it may constantly change as the market changes, meaning the knowledge must be reacquired. So, the effort, schedule, staff and cost may increase or decrease without respect to system size at all.

Almost all estimation processes and tools provide a way of *tuning* the final-size-driven estimate by adjusting parameters which represent the systems' attributes. We also trust these adjustments will track to the effort necessary to acquire the knowledge. The net result of this tuning may entirely submerge the effect of the system size.

- **The effort-time relationship is not linear.** In fact, it is a high order reciprocal exponent [4]. It is common for organizations to believe that the process of building software is, well, a building process rather than a knowledge acquiring process. Accordingly, they operate on a set of assumptions based upon manufacturing. This includes the relatively linear relationship of effort (people, machines) to time to deliver. In a factory, if we double the number of machines or run the machines twice as long or twice as fast, we will approximately double the output. This math simply does not apply to software because it is not a manufacturing process. It also partly explains why adding people to a project – particularly when it is already running late and the schedule is already compressed – is not an effective tactic (see Figure 3).

Figure 3: *High Order Reciprocal Power*



The Job of Estimation

The real job of estimation is not what it seems. True, it does have a very important role in determining the basic planning parameters of staff size, project duration, effort and cost (and for some estimation models, quality and defects). This is the classical *tell me when the project will be done* role of estimation.

The calibrations necessary to achieve a useful, as opposed to accurate, answer from an estimate are complex. They *characterize* things: the system being built, the environment and people working on the project, the management of that environment, the tools and their effectiveness, the level of documentation, and many other factors. This is clearly seen in the operation of many parametric estimation tools and processes. For instance, the COCOMO II model uses Scale Factors which determine the size exponent. These include the following:

- System/project attributes: How new the project is.
- Project/process attributes: Development process flexibility and architecture risk resolution.
- Team attributes: Team cohesion.
- Organizational attributes: Process maturity [5].

COCOMO II further expands on its characterization with a set of *cost driver* factors which reflect everything from the documentation level to the *complexity* of the system being built.

Given a reasonably sound characterization of an environment and system using these types of factors – a very significant task in itself – an estimation process becomes an analogue of the projects being run in that environment.

A Certain Uncertainty

Even if we think we are able to *accurately* size a projected system, and we have good data which we think characterizes the team, sys-

tem, and environment, we invariably find that these are not certain. With each factor, there comes some degree of variability: We may have been *this* productive in the past, but our productivity may be different now. The project might be *this* big, but it could be that big. It could be on the simple side of complex, or the complex side of simple. The new tools or language we are planning to use might help a lot, a little, or they might require more effort than they save. The only way to be truly certain is to try it.

For each of the factors, we can postulate that they will likely operate over a range of values. The product of all these variances determines the aggregate uncertainty of the project and defines the slope of the cumulative probability S-curve in Figure 2. There are many challenges to calculating these ranges, not the least being that the factors are not independent. Processing the individual variances in a statistically legitimate way allows us to calculate the *total uncertainty* in the final project solution(s).

Estimation as Simulation

Here we get to the real purpose of estimation. If we have reasonably characterized the environment, if we have established some operational variance for the size of the system and its complexity, if we have some idea of the ranges of difficulty in obtaining the knowledge for the system, and if we have some calibrated way of processing this information, we could *simulate* what might happen when we run the project.

The concept of a statistical approach to the management of software (especially under the umbrella of Six Sigma) has its detractors and they have some very good points [6]. Developing software is not the repetitive cranking out of identical units. Indeed, doing something differently from the last time is a bad thing from a manufacturing perspective, and most of the effort in statistical process control is dedicated to identifying, analyzing, and removing variance. But in software, variance is the reason why we have a project at all – if we wanted to do it just like the last time, we would simply use whatever we produced last time. Again, software is not a product at all, it is a medium in which we express, store, and make active the knowledge that we gain when we run a project. It is the knowledge that is the thing, the software is simply where we put it.

We do not have the resources to run the same project multiple times to see how to run it best. We only get to run a project once. However, we could set up an estimation system, with certified and controlled inputs, that reasonably collects the likely variances in the key characterizing factors of product, personnel, technology, and environment, and we could model the interaction of these variables reasonably well. Doing this, we can simulate the behavior of our organization when it runs the project under a given set of conditions. These results can only be expressed in probabilistic terms – since we have uncertain inputs, we must have uncertain outputs. These outputs look a lot like statistical variance analyses, even though we only have one project. Many companies have and use various estimation processes and tools, but few have established estimation *systems* that control, audit, and report project estimation and risk data in the same way we control, audit, and report on accounting data.

The financial management sections in most companies have financial models that they create, manage, and use. These models incorporate key factors in the financial markets: inflation, growth in Gross National Product, cost of capital, market sensitivity, etc. Companies find these tools very valuable in helping to understand what kinds of decisions might be more optimal than others. Do these tools predict the future? No. They cannot do that. But they can and do help in the financial management of companies; they are very valuable tools and systems. We could do the same thing for software projects and estimation.

Comedian Woody Allen once remarked "the only thing I cannot accurately predict is the future…". What an estimation simulator could do is help identify more (or less) optimal decisions about how we run our projects, before we actually run them. We often teach pilots on simulators. They do not replace learning on the real thing, but you can try things and test out behaviors on a simulator that you would not want to try on the real thing.

Fifteen Too Many, One Too Few

Several months ago, a client of mine was considering implementing a large project in 15 equal increments spread over three years. Using estimation tools to model the whole system in a one-release, three-year delivery, *big bang* approach we showed that this was not a highly constrained system. However, modeling the 15 increments in our estimation system showed that the sum of the parts was a lot bigger than the whole. We could demonstrate that the overlapping increments inserted a very high degree of risk into the project that would only become evident some way down the line. This project would look pretty good for about two years and then the wheel would fall off. The big bang approach was much less risky, but the customer would see no value for three years. We modeled many possible solutions including a four, unequal-increment solution that we were able to demonstrate would deliver the most functionality to the customer at the earliest date, with the lowest risk.

We could have learned that the 15 increment solution was a bad idea by trying it, thereby costing the company several million dollars. Or, we could learn the same lesson by simulating what would happen and preemptively picking a more reasonable course.

There is little doubt we need to improve our performance in software project estimation. The stakes are very high, but if we align our expectations of estimation in accordance with the reality of software development and set up our organizations to feed a software development business simulation system, the rewards will be even higher. We can do that.◆

References

1. Wood, Michael. The Road to Delphi: Scenes From the History of Oracles Farrar. New York: Straus and Giroux, 2003.
2. Armour, P.G. "The Case for a New Business Model." Communications of the ACM 43.8 (2000): 19-22.
3. Armour, P.G. "The Laws of Software Process." Boca Raton, FL: Auerbach Publishers, 2003.
4. Putnam, Lawrence H., and Ware Myers. Measures for Excellence. Englewood Cliffs, NJ: Yourdon Press/Prentice Hall, 1992.
5. Boehm, Barry, W, et al. "Software Cost Estimation with COCOMO II." Upper Saddle River, NJ: Prentice Hall, 2000.
6. Binder, Robert V. "Can a Manufacturing Quality Model Work for Software?" IEEE Software 14.5 (1997): 101-105.

About the Author

Phillip G. Armour is a senior consultant for Corvus International, Inc. He is a contributing editor at *Communications of the ACM* and authored the book "The Laws of Software Process."

Corvus International, Inc.
205 Briargate LN
Deer Park, IL 60010
Phone: (847) 438-1609
E-mail: armour@corvusintl.com

Science Fair, Farce, and Free-For-All

I started this article on January 21st, the most depressing day of the year [1]. Dr. Cliff Arnall gauged the third Monday of January to be the most depressing day using a formula based on weather, holiday debt, and failed resolutions.

The exception to Arnall's formula involves engineers with children in kindergarten through eighth grade. The brightly colored notice they receive each year for the school science fair offers a respite from depression. The prospect to tinker with science offsets the depressing effects of weather, debt, and failed resolutions.

Let's be honest: Most engineers engage in little or no engineering. They entered engineering to design and build but the dirty little secret they don't tell you in college is that only a small percentage of engineers actually design or build. Most engineers document, configure, test, meet, review, manage, meet, inspect, and meet again, but few design. Those who do design rarely get hands-on building projects and hands-on software is non-existent (note: software builds and keyboard strokes do not count).

When that science fair paper hits home, the pent-up frustration of deprived engineers uncorks like a potato out of a butane fueled polyvinyl chloride pipe – another release activity for hamstrung engineers. Wheels start turning, the engineering paper comes out, and the Home Depot account mounts.

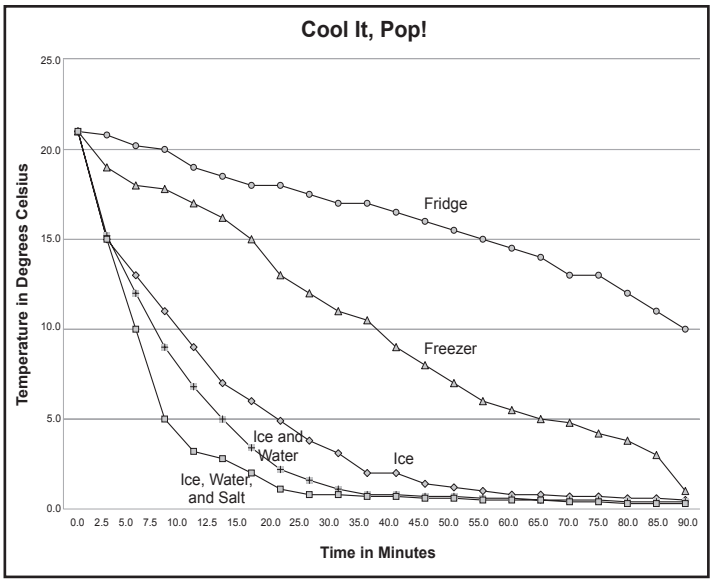
Now, I'm a big fan of parental involvement in school activities, and I also support alleviation of engineering frustration, but I must caution my fellow engineers: Do not overdo it. Here are signs your child may not be getting the expected science project experience:

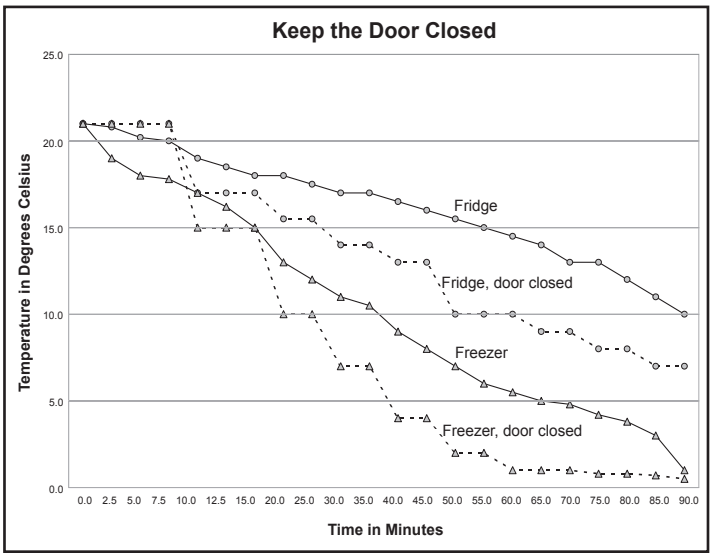
- The project takes more than 20 minutes to set up.
- Armed guards are required to protect the project.
- You ask the janitor for a high voltage outlet.
- Lloyds of London insures the project.
- You have to return the derrick crane by 4:00 p.m.
- Occupational Safety and Health Administration inspection is required.
- Your child answers all questions with, "Mom!"

Keep your project simple, involve your child, and, above all, please leave the volcanoes, Mentos geysers, cake baking instructions, and soda pop-soaked teeth at home.

My daughter, Hannah, was inspired by MythBusters to determine the fastest way to cool a can of soda pop [2]. The results, in the Cool It Pop graph, determined that a cooler full of ice and salt water is your best bet – provided you don't have a fire extinguisher on hand.

Hannah noticed the refrigerator and freezer seemed very slow to cool. I noticed Hannah left the door open during measures, diminishing the refrigerator's cooling ability. She modified her measurement by pulling the can out of the refrigerator, closing the door during measurement, and then returning it. The results were much better as displayed in the Keep the Door Closed graph.





What can the science fair teach us about tracking engineering projects? First and foremost, if you manage engineers, give them at least one task requiring designing, tinkering, or building. This act alone will save both yours and the engineer's sanity.

Second, resist the temptation to over-measure. If measurements become more important than the project itself, your measures will be sullied. Keep the door closed and let your engineers engineer.

Third, unbiased and precise measures are impossible. Factor that into your analysis, be tolerant on precision, and vigilant on accuracy (see [3] to discern the difference).

Finally, waste not, want not – keep it simple. Not simple minded; simple to implement, simple to measure, simple to use, and simply effective.

—Gary A. Petersen
Arrowpoint Solutions, Inc.
gpetersen@arrowpoint.us

References

1. Tanker, Bill. "The Most Dangerous Day of the Year." Time Magazine Jan. 2008.
2. Mythbusters. Discovery Channel <http://dsc.discovery.com/fansites/mythbusters/mythbusters.html>.
3. Petersen, Gary A. "Ready, Fire, Aim." CrossTalk Sept. 2006.

CrossTalk / 517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737

CrossTalk is co-sponsored by the following organizations:

Homeland Security